Matematická logika a jej aplikácie

Mathematical Logic and its Applications

Martin Belanec

Bakalárska práca
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin BELANEC**
Osobní číslo:	**A10107**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Matematická logika a její aplikace**

Zásady pro vypracování:

1. **Vysvětlete pojem matematická logika a ostatní pojmy týkající se tzv. dvouhodnotové logiky.**
2. **Vysvětlete základní pojmy vícehodnotové logiky, fuzzy logiky, případně další a jejich základní aplikace.**
3. **Popište příkazy softwaru Mathematica týkající se matematické logiky a jejich využití při řešení úloh z matematiky.**
4. **Uveďte využití logických operátorů při vytváření logických obvodů.**
5. **Popište využití logických operátorů při vytváření databázových systémů.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KVASNIČKA, Vladimír a Jiří POSPÍCHAL. Matematická logika. Bratislava: STU vydavateľstvo, 2007. ISBN 8022724491**
2. **OSTRAVSKÝ, Jan a Vladimír POLÁŠEK. Diferenciální a integrální počet funkce jedné proměnné: vybrané statě. Zlín: Univerzita Tomáše Bati ve Zlíně, 2011. ISBN 978-80-7454-124-7.**
3. **VRBA, R., R. KUCHTA, O. SAJDL, P. HUB, M. SKOČDOPOLE, L. FUJCIK, J. HÁZE, M. ZEMÁNEK a R. VRBA. Multimediální učebnice digitálních obvodů. Brno: FEKT VUT, 2004. Dostupné z: http://www.umel.feec.vutbr.cz/bdom**
4. **KAPRÁLIK, P., GALANOVÁ, J., POLAKOVIČ, M. Logické systémy. Bratislava: STU vydavateľstvo, 2009. ISBN 9788022732055**
5. **ŠEDA, Miloš. Databázové systémy. Brno: FSI VUT, 2002. Dostupné z: www.uai.fme.vutbr.cz/ mseda/DBS02_BS.pdf**

Vedoucí bakalářské práce: **Mgr. Vladimír Polášek, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **24. února 2013**

Termín odevzdání bakalářské práce: **14. června 2013**

Ve Zlíně dne 24. února 2013

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Táto bakalárska práca popisuje použitie matematickej logiky pri riešení matematických úloh, vytváraní logických obvodov a dopytovania v databázových systémoch. V teoretickej časti sú vysvetlené pojmy týkajúce sa dvojhodnotovej logiky a taktiež základné pojmy viachodnotových logík. V praktickej časti sa práca zaoberá využitím týchto logík vo vyššie popísaných okruhoch. Taktiež obsahuje praktické ukážky riešení logických matematických úloh v softwari Wolfram Mathematica, zostavovania logických obvodov v softwari Logisim a zápisu dopytovacích príkazov v softwari Microsoft SQL Server.

Kľúčové slova: matematická logika, dvojhodnotová logika, fuzzy logika, Wolfram Mathematica, logický obvod, Logisim, databázový systém, Microsoft SQL Server

ABSTRACT

This bachelor work describes the use of mathematical logic in solving mathematical problems, creating the logic circuits and querying in database systems. In the theoretical section, there are explained the concepts of bivalent logic as well as basic concepts of multivalent logics. In the practical part, the bachelor work deals with the use of these logics in the above-described circuits. It also includes practical demonstrations of mathematical logic solving tasks in the software Wolfram Mathematica as well as the compilation of logic circuits in the software Logisim and writing the querying commands in the software Microsoft SQL Server.

Keywords: Mathematical logic, bivalent logic, fuzzy logic, Wolfram Mathematica, logic circuit, Logisim, database system, Microsoft SQL Server

Na tomto mieste by som sa chcel poďakovať vedúcemu bakalárskej práce pánovi Mgr. Vladimíru Poláškovi, Ph.D. za odbornú pomoc, vynaložený čas, cenné rady a v neposlednom rade za trpezlivosť, ktorú so mnou mal. Ďalej by som sa chcel taktiež poďakovať rodičom, súrodencom a priateľke za podporu behom štúdia.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronicky nahraná do IS/STAG jsou totožné.

Ve Zlíně ……..……………….

podpis diplomanta

OBSAH

ÚVOD

Matematická logika je oblasť matematiky, ktorá je úzko spojená s informatikou. Jej uplatnenie nájdeme v mnohých systémoch ako sú logické obvody, databázové systémy, regulačné obvody a mnohé iné. Cieľom tejto bakalárskej práce je poskytnúť študentom ucelený text, ktorý im zvládne lepšie pochopiť rôzne pojmy matematickej logiky a ich uplatnenie v reálnych systémoch a aplikáciách.

Práca je rozdelená na dve časti, teoretickú a praktickú. V teoretickej časti sú popísané základné vlastnosti rôznych logík matematickej logiky a v praktickej časti využitie tejto oblasti matematiky v rôznych aplikáciách. Pri každom novom pojme je uvedený názorný príklad, ktorý umožní rýchlejšie pochopiť danú problematiku.

Teoretická časť obsahuje kapitoly 1 až 7. Kapitola 1 predstavuje úvod do matematickej logiky, oboznamuje s jej históriou a jej základným rozdelením. Kapitoly 2 a 3 prezentujú klasickú logiku. Je v nich bližšie popísaná výroková logika a jej algebra. V kapitolách 4 a 5 nastáva odklon od klasických logík a sú tu rozobraté základné pojmy neklasických logík, ako je Łukasiewiczova a Fuzzy logika, taktiež je tu možné sa dozvedieť o ich základnej aplikácii. Kapitoly 6 a 7 len stručne oboznamujú s logickými obvodmi a databázovými systémami, pretože v praktickej časti je podrobnejšie vysvetlené využitie matematickej logiky v týchto systémoch.

Praktickou časťou sa zaoberajú kapitoly 8 až 10, v ktorých sa nachádza využitie logík z teoretickej časti v matematických príkladoch, riešených v softwari Wolfram Mathematica, pri vytváraní logických obvodov a dopytov v databázových systémoch.

I. TEORETICKÁ ČASŤ

1 LOGIKA

Logika je filozofická disciplína, ktorá sa zaoberá popisom správnosti usudzovania. V logike nejde o to, aké má tvrdenie obsah, ale o to, aby malo správnu formu bez ohľadu na jeho pravdivosť alebo nepravdivosť. [1]

1.1 Matematická logika

Logika študuje všeobecné formy usudzovania na symbolickej úrovni, v ktorej sa ignoruje konkrétny obsah jednotlivých tvrdení. Preto býva moderná logika označovaná ako matematická logika. Základným prvkom v oboch prípadoch je používanie symbolov a ich spájanie pomocou logických spojok do väčších celkov, ale aj formalizácia procesu transformácie daných väčších celkov na iné, metódami, ktoré sú charakteristické pre matematiku. [2]

1.2 História logiky

Za zakladateľa logiky je považovaný Aristoteles (384 – 322 p. n. l.), ktorý systematicky popísal niektoré typy uvažovania a oddelil správne úvahy od nesprávnych. Súčasne sa Eukleidés z Megar zaoberal skúmaním myslenia a určil axiomatický systém. Pokračovatelia boli stoikovia, hlavne Zenón a Chrýsippos. Symbolická logika sa začala rozvíjať až v 19. storočí. Hlavnou príčinou bolo skúmanie logických paradoxov. Z logiky boli odstránené slová z bežného jazyka a tým získala formálny charakter. Medzi najvýznamnejších predstaviteľov logiky patria: B. Bolzano, G. Boole, A. de Morgan, G. Frege, G. Peano, D. Hilbert, B. Russell, A. Tarski, T. Skolem, A. Church, J. Herbrand, A. Turing, G. Gentzen a K. Gödel. [3]

1.3 Rozdelenie logík

1.3.1 Klasická logika

Pod klasickou logikou sa rozumie logika, v ktorej sa predpokladá, že pravdivosť daného tvrdenia môže nadobudnúť dve hodnoty. Tvrdenie môže byť buď pravdivé alebo nepravdivé. Tvrdenia môžu tvoriť rôzne zložitejšie celky, ktoré spájame spojkami („nie je pravda, že...“, „...a...“, „...alebo...“, „ak...,potom...“, atď.) a kvantifikátormi („existuje *x*...“,

„pre každé *x*...“), kde pravdivostné ohodnotenie zložených tvrdení závisí na pravdivostných hodnotách skladaných tvrdení a použitých spojkách. [2]

V tejto bakalárskej práci je popísaná výroková logika, ktorá tvorí základ klasickej logiky a jej algebra.

1.3.2 Neklasická logika

Neklasické logiky sa odlišujú od klasických tým, že tvrdenia nemusia byť len pravdivé alebo nepravdivé, ale môžu mať aj iné pravdivostné ohodnotenia. Môžu taktiež používať logické spojky, ktoré nie sú používané v klasickej logike alebo môžu mať iný význam. Tieto nové spojky môžu určovať časové či modálne aspekty tvrdení, alebo môžu spájať tri tvrdenia do nového, zložitejšieho. [4]

V tejto práci sú popísané dva typy neklasických logík a to sú Łukasiewiczova logika a Fuzzy logika.

2 VÝROKOVÁ LOGIKA

Výroková logika študuje formy usudzovania, ktorých platnosť záverov závisí od toho, či sú tvrdenia pravdivé alebo naopak nepravdivé. [4]

2.1 Výrok

Elementárny výrok je jednoduchá oznamovacia veta, u ktorej sa môže rozhodnúť, či je tvrdenie pravdivé alebo nepravdivé, pričom môže vždy nastať len jedna z týchto možností. [4]

2.1.1 Výroková premenná

Výrokové premenné sa používajú na zovšeobecnenie výrokov. Teda výroková premenná je výraz, ktorý môže reprezentovať ľubovoľný výrok. Výrokové premenné sa označujú malými písmenami *p*, *q*, *r*, ... [1]

2.1.2 Pravdivostná hodnota

Výrok, ktorý vyjadruje platné tvrdenie, je pravdivý. Ak vyjadruje opak, tak nepravdivý. Pravda a nepravda sa nazývajú pravdivostnými hodnotami. 0 a 1 reprezentujú tieto pravdivostné hodnoty. Pravdivostné ohodnotenie sa dá považovať za ľubovoľné zobrazenie výrokových premenných daného jazyka výrokovej logiky do množiny $\{0,1\}$. Pravdivostná hodnota výroku sa označuje $e(p)$. [2]

2.2 Logické spojky

Logické spojky sú operátory, pomocou ktorých sa spájajú elementárne výroky do zložitejších zložených výrokov. Pravdivostná hodnota týchto zložených výrokov je určená na základe týchto elementárnych výrokov a tiež na konkrétnej logickej spojke. Logické spojky sa delia na unárne a binárne. Unárne spojky vykonávajú operáciu len nad jedným výrokom. Ich celkový počet je 4. Naopak binárne spojky spájajú dva výroky v jeden zložený. Ich celkový počet je 16. Vo výrokovej logike používame len jednu základnú unárnu spojku (negáciu) a štyri základné binárne spojky (konjunkciu, disjunkciu, implikáciu, ekvivalenciu). [2]

2.2.1 Negácia

Táto unárna logická spojka má slovné vyjadrenie „nie je pravda, že...“. Zapisuje sa pomocou znaku $\neg$. Negácia pracuje s výrokmi tak, že mení pravdivý výrok na nepravdivý a naopak, nepravdivý na pravdivý, čo je možné vidieť aj v nasledujúcej tabuľke Tab. 1. [4]

p	$\neg p$
0	1
1	0

Tab. 1: Negácia

Príklad 1: Výrok *p*: Praha je hlavné mesto Českej republiky.

Negácia: Praha **nie** je hlavné mesto Českej republiky.

Nie je pravda, že Praha je hlavné mesto Českej republiky.

2.2.2 Konjunkcia

Binárna spojka konjunkcie spája dva výroky v jeden zložitejší výrok pomocou spojky „a“. Formálne sa označuje pomocou znaku $\wedge$, teda zápis konjunkcie výrokov p a q sa zapisuje „ $p \wedge q$ “. Táto spojka je symetrická, teda nezáleží na poradí výrokov „$(p \wedge q) = (q \wedge p)$“. Zložený výrok spojený pomocou konjunkcie je pravdivý len vtedy, ak sú pravdivé obidva výroky p, q. [4]

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 2: Konjunkcia

V Tab. 2 je zobrazené pravdivostné ohodnotenie konjunkcie dvoch výrokov.

Príklad 2: Výrok *p*: Číslo 255 je deliteľné piatimi.

Výrok *q*: Súčet číslic čísla 255 je 12.

Konjunkcia: Číslo 255 je deliteľné piatimi **a** súčet číslic čísla 255 je 12.

V Pr. 2 môžeme vidieť, že oba výroky p aj q sú pravdivé. Pravdivostné ohodnotenie tohto zloženého výroku pomocou logickej spojky konjunkcie je taktiež pravdivé.

2.2.3 Disjunkcia

Binárna spojka disjunkcie spája dva výroky v jeden zložitejší pomocou spojky „alebo“. Formálne sa označuje pomocou znaku $\vee$. Zápis disjunkcie výrokov je teda nasledovný „$p \vee q$“. Tak ako pri konjunkcii, aj táto spojka je symetrická, čiže platí „$(p \vee q) = (q \vee p)$“. Takto zložený výrok pomocou spojky „alebo“ je pravdivý práve vtedy, ak aspoň jeden výrok z p, q je pravdivý. Z toho vyplýva, že disjunkcia nie je vo forme vylučujúceho „alebo“. Pravdivostné ohodnotenie je zobrazené v tabuľke Tab. 3. [4]

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 3: Disjunkcia

Príklad 3: Výrok *p*: Auto má predný náhon.

Výrok *q*: Auto má zadný náhon.

Disjunkcia: Auto má predný **alebo** zadný náhon.

Pr. 3 znázorňuje použitie spojky disjunkcie. Z výrokov p a q môžeme usúdiť, že zložený výrok bude vždy pravdivý, lebo auto musí mať buď predný alebo zadný náhon. V špeciálnom prípade môže mať auto náhon na všetky štyri kolesá. V tomto prípade by obidva elementárne výroky boli pravdivé, pretože auto by malo aj predný aj zadný náhon, teda zložený výrok by bol taktiež pravdivý.

2.2.4 Implikácia

Binárna spojka implikácie spája dva výroky v zložitejší pomocou spojok „ak..., potom...“. Formálne označenie spojky je pomocou znaku ⇒, teda zápis spojenia výrokov je „ $p \Rightarrow q$ “. Pri tejto spojke záleží na poradí, čiže nie je symetrická. Zložený výrok pomocou implikácie je nepravdivý len vtedy, ak prvý výrok je pravdivý a druhý nepravdivý. Pre ostatné prípady je implikácia pravdivá, čo je zobrazené v tabuľke Tab. 4. [4]

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Tab. 4: Implikácia

Príklad 4: Výrok *p*: Nosím dioptrické okuliare.
Výrok *q*: Mám poškodený zrak.
Implikácia: **Ak** nosím dioptrické okuliare**, potom** mám poškodený zrak.

Ak v Pr. 4 predpokladáme výrok *p* ako pravdivý, a naopak výrok *q* ako nepravdivý, pre zložený výrok pomocou logickej spojky implikácie by to v takomto príklade znamenalo, že jeho celkové pravdivostné ohodnotenie by bolo nepravdivé.

2.2.5 Ekvivalencia

Binárna spojka ekvivalencie spája dva výroky v zložitejší výrok pomocou spojky „... práve vtedy, keď ...“. Označenie spojky je pomocou znakov ≡ alebo ⇔, zápis je teda nasledovný „ $p \equiv q$ “. Tak ako pri konjunkcii a disjunkcii, aj táto spojka je symetrická, čiže „$(p \equiv q) = (q \equiv p)$“. Zložený výrok je pravdivý vtedy a len vtedy, ak sú výroky buď súčasne pravdivé alebo nepravdivé. Túto skutočnosť zobrazuje nasledujúca tabuľka Tab. 5. [4]

p	q	$p \equiv q$
0	0	1
0	1	0
1	0	0
1	1	1

Tab. 5: Ekvivalencia

Príklad 5: Výrok p: Číslo 24 je párne.
Výrok q: Číslo 24 je deliteľné dvomi.
Ekvivalencia: Číslo 24 je párne **práve vtedy a len vtedy ak** je deliteľné dvomi.

Z definície ekvivalencie vieme, že zložený výrok je pravdivý práve vtedy, ak obidva jeho výroky majú rovnaké pravdivostné ohodnotenie. Z Pr. 5 môžeme usudzovať, že elementárne výroky p a q sú pravdivé, teda zložený výrok má v takomto prípade pravdivé pravdivostné ohodnotenie.

2.2.6 Ostatné logické spojky

Ostatné logické spojky nie sú vo výrokovej logike veľmi používané, ale niektoré z nich majú veľký význam pri reálnych logických aplikáciách a sú vysvetlené v kapitole 3 Boolova algebra.

Medzi všetkými spojkami existujú vzťahy, ktorými môžeme jednotlivé spojky vyjadriť pomocou negácie a konjunkcie.

Príklad 6: Disjunkcia vyjadrená pomocou negácie a konjunkcie:

$$p \vee q \equiv \neg[\neg(p \vee q)] \equiv \neg[\neg p \wedge \neg q]$$

Implikácia vyjadrená pomocou negácie a konjunkcie:

$$p \Rightarrow q \equiv \neg[p \wedge \neg q]$$

2.3 Formula výrokovej logiky

Pod pojmom formula výrokovej logiky rozumieme:

- každú výrokovú premennú $p, q, r, \ldots,$
- výrokové konštanty P a N, kde P označuje výrok pravdivý a N výrok nepravdivý,
- ak sú výrazy α a β výrokové formule, potom sú aj $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$, $(\alpha \equiv \beta)$ výrokové formule,
- iné výrazy výrokovými formulami nie sú.

Výrokové formule o n výrokových premenných zapisujeme ako $\alpha(p_1, p_2, \ldots, p_n)$. [5]

2.3.1 Tautológia

Tautológiou je každá formula, ktorá pre ľubovoľnú kombináciu pravdivostných hodnôt výrokových premenných nadobúda hodnotu 1, čiže je vždy pravdivá. [1]

Príklad 7: Výrok p: Sneží.
Výroková formula: p alebo nie je pravda, že p
Symbolický zápis: $\alpha(p) = \alpha[p \vee (\neg p)]$

1	2	3
p	$\neg p$	$p \vee (\neg p)$
0	1	1
1	0	1

Tab. 6: Pravdivostné ohodnotenie formule α

2.3.2 Kontradikcia

Kontradikciou je každá formula, ktorá pre ľubovoľnú kombináciu pravdivostných hodnôt výrokových premenných nadobúda hodnotu 0, takže je vždy nepravdivá. [1]

Príklad 8: Výrok p: Sneží.

Výroková formula: p a nie je pravda, že p

Symbolický zápis: $\beta(p) = \beta[p \wedge (\neg p)]$

1	2	3
p	$\neg p$	$p \wedge (\neg p)$
0	1	0
1	0	0

Tab. 7: Pravdivostné ohodnotenie formule β

2.3.3 Splniteľnosť

Formula je splniteľná vtedy a len vtedy, ak je aspoň pre jednu ľubovoľnú kombináciu pravdivostných hodnôt výrokových premenných pravdivá. Teda splniteľnou formulou je aj každá tautológia. [1]

Príklad 9: Výroková formula: $\gamma(p,q) = \gamma[(p \equiv q) \wedge (\neg p)]$

1	2	3	4	5
p	q	$p \equiv q$	$\neg p$	$3 \wedge 4$
0	0	1	1	1
0	1	0	1	0
1	0	0	0	0
1	1	1	0	0

Tab. 8: Pravdivostné ohodnotenie formule γ

2.4 Kvantifikované výroky

2.4.1 Výroková forma

Výroková forma je výraz, ktorý obsahuje aspoň jednu premennú a sú k nej priradené dve množiny, t.j. definičný obor výrokovej formy a obor pravdivosti výrokovej formy. Definičný obor výrokovej formy je množina všetkých prípustných hodnôt, pri ktorých sa po dosadení za premenné z výrokovej formy stane výrok. Obor pravdivosti výrokovej formy je množina všetkých prípustných hodnôt, pre ktoré je výrok pravdivý. Výrokovú formu označujeme veľkými písmenami *A*, *B*, ..., *Z* [6]

Príklad 10: Výroková forma $A(x)$: $x^2 \geq 100$

Definičný obor: $x \in R$

Výrok: $x^2 \geq 100,\ x \in R$

$A(x)$ predstavuje výrokovú formu s premennou x, ktorá spolu s definičným oborom R (reálne čísla) tvorí výrok.

2.4.2 Kvantifikácia výroku

Výrok je možné vytvoriť z výrokovej formy aj bez dosadenia hodnôt za premenné. V takomto prípade musíme výrok kvantifikovať, čo znamená určiť množinu prípustných hodnôt. Pre kvantifikovanie výrokov používame existenčný a všeobecný kvantifikátor. [6]

Existenčný kvantifikátor sa označuje $\exists$. Pomocou neho sa dá zjednodušene zapísať „Existuje x, že platí $A(x)$.“, kde $A(x)$ predstavuje výrokovú formu a x premennú s určitým definičným oborom. Tento kvantifikovaný výrok sa môže zapísať ako

$$\exists x\ A(x) \quad [6]$$

Pomocou výrokovej logiky sa existenčný kvantifikátor zapisuje ako

$$\exists x\ A(x) = A(x_1) \vee A(x_2) \vee ... \vee A(x_n)$$

$x_1, x_2, ..., x_n$ – množina všetkých prípustných hodnôt [4]

Všeobecný kvantifikátor sa označuje $\forall$. Pomocou neho zapisujeme zjednodušene výrok „Pre všetky x platí $A(x)$.“, kde $A(x)$ taktiež predstavuje výrokovú formu a x premennú s určitým definičným oborom. Všeobecný kvantifikátor sa môže zapísať ako

$$\forall x\ A(x)\ [6]$$

Pomocou výrokovej logiky sa všeobecný kvantifikátor zapisuje ako

$$\forall x\ A(x) = A(x_1) \wedge A(x_2) \wedge \ldots \wedge A(x_n)$$

$x_1, x_2, \ldots, x_n$ – množina všetkých prípustných hodnôt [4]

Príklad 11: Výroková forma $A(x)$: x má právo na život

Definičný obor D: množina ľudí

Kvantifikovaný výrok: Každý človek má právo na život.

Zápis: $\forall x \in D\ A(x)$

2.4.3 Negácie kvantifikovaných výrokov

p	$\neg p$
$\exists x \in D\ A(x)$	$\neg[\exists x \in D\ A(x)] \equiv \forall x \in D\ \neg A(x)$
$\forall x \in D\ A(x)$	$\neg[\forall x \in D\ A(x)] \equiv \exists x \in D\ \neg A(x)$

Tab. 9: Negácia kvantifikovaných výrokov

Príklad 12: Negácia výroku z Pr. 11: Existuje osoba, ktorá nemá právo na život.

Zápis: $\exists x \in D\ \neg A(x)$

3 BOOLOVA ALGEBRA

Boolova algebra je náuka o operáciách na množine, ktorá obsahuje logické hodnoty 0 a 1, logické premenné, a na ktorej sa používajú tieto základné operácie:

- logický súčin – označuje sa operátorom $\wedge, a, AND$ alebo symbolom $\cdot$,
- logický súčet – označuje sa operátorom $\vee, alebo, OR$ alebo symbolom $+$,
- negácia – označuje sa symbolom $\neg$ pred operandom alebo vodorovnou čiarou nad operandom.

Boolova algebra sa označuje ako algebra stavov. Vyplýva to z toho, že premenná môže nadobudnúť len dve možné hodnoty, a to 0 alebo 1. Tieto logické hodnoty v analógii s výrokovou logikou predstavujú výrokovú nepravdivú konštantu a výrokovú pravdivú konštantu. [7]

3.1 Boolova funkcia

Pravdivostné ohodnotenie formule výrokovej logiky $\alpha(x_1, x_2, ..., x_n)$ obsahujúcej n výrokových premenných $x_1, x_2, ..., x_n$ je interpretované ako Boolova funkcia, ktorá priradí n logickým premenným logickú funkčnú hodnotu 0 alebo 1. [4]

$$y = \alpha(x_1, x_2, ..., x_n)$$

3.1.1 Unárne Boolove funkcie

x	u_1	u_2	u_3	u_4
0	0	0	1	1
1	0	1	0	1

Tab. 10: Unárne Boolove funkcie

Tabuľka unárnych Boolových funkcií zobrazuje všetky možné funkcie jednej premennej. Tretia unárna funkcia u_3 predstavuje výrokovú spojku negácie. Ostatné tri unárne funkcie nemajú svojich reprezentantov vo výrokovej logike. [4]

3.1.2 Základné binárne Boolove funkcie

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	1	1	0	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0

Tab. 11: Binárne Boolove funkcie

Celkový počet binárnych Boolových funkcií je $16 = 2^4$. Funkcia f_1 sa nazýva logický súčin a reprezentuje logickú spojku konjunkcie výrokovej logiky. Funkcia f_2 zobrazuje logický súčet a reprezentuje logickú spojku disjunkcie výrokovej logiky. Funkcie f_3 a f_4 predstavujú logické spojky implikácie a ekvivalencie. Ďalšie tu znázornené logické funkcie nemajú svojich reprezentantov vo výrokovej logike, ale využívajú sa v elektronických aplikáciách Boolovej teórie. Funkcia f_5 sa nazýva exkluzívny súčet $(Xor, \oplus)$ a predstavuje negáciu ekvivalencie. Funkcia f_6 označuje Shefferov symbol $(Nand, \uparrow)$, je to negácia konjunkcie. Funkcia f_7 predstavuje Peircov symbol $(Nor, \downarrow)$, čo je negácia disjunkcie. Pomocou Peircova a Shefferova symbolu sa dá vyjadriť každá unárna aj binárna funkcia. [7]

3.2 Špeciálne tvary Boolových funkcií

Každá výroková formula môže byť prepísaná do špeciálneho tvaru Boolovej funkcie. Tieto špeciálne tvary sú disjunktívny normálny tvar (DNF) a konjunktívny normálny tvar (CNF).

Funkcia zapísaná v disjunktívnom normálnom tvare sa získa ako súčet základných súčinov priamych alebo negovaných premenných. Základné súčiny sa zapisujú len pre tie kombinácie premenných, pri ktorých výsledná funkcia y nadobúda hodnotu 1, tzn., že logický súčin sa zapíše pre všetky výrokové premenné, pri ktorom je pravdivostné ohodnotenie formule pravdivé. Premenné, ktoré nadobúdajú hodnotu 1, sa zapisujú ako priame premenné, čiže bez operátora negácie. Naopak tie, ktoré nadobúdajú hodnotu 0, sa zapisujú ako negované premenné, tzn. s operátorom negácie.

Funkcia zapísaná v konjunktívnom normálnom tvare sa získa ako súčin základných súčtov priamych alebo negovaných premenných. Základné súčty sa zapisujú len pre tie kombinácie premenných, pri ktorých výsledná funkcia y nadobúda hodnotu 0. Premenné, ktoré nadobúdajú hodnotu 0, sa zapisujú ako priame premenné. Naopak tie, ktoré nadobúdajú hodnotu 1, sa zapisujú ako negované premenné. [8]

Príklad 13: Výroková formula: $\alpha(p,q)=\alpha[(p\equiv q)\wedge(\neg p)]$

Pravdivostné ohodnotenie formule:

p	q	$(p\equiv q)\wedge(\neg p)$	**Základný súčin**	**Základný súčet**
0	0	1	$(\neg p)\wedge(\neg q)$	$p\vee q$
0	1	0	$(\neg p)\wedge q$	$p\vee(\neg q)$
1	0	0	$p\wedge(\neg q)$	$(\neg p)\vee q$
1	1	0	$p\wedge q$	$(\neg p)\vee(\neg q)$

Tab. 12: Základné súčiny a súčty formule α

DNF: $\alpha(p,q)=(\neg p)\wedge(\neg q)$

Ekvivalentný prepis: $y=\overline{x_1}\cdot\overline{x_2}$

CNF: $\alpha(p,q)=[p\vee(\neg q)]\wedge[(\neg p)\vee q]\wedge[(\neg p)\vee(\neg q)]$

Ekvivalentný prepis: $y=(x_1+\overline{x_2})\cdot(\overline{x_1}+x_2)\cdot(\overline{x_1}+\overline{x_2})$

3.3 Základné vzťahy Boolovej algebry

Logický súčin	Logický súčet	Zákon
$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$	komutatívny
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$	asociatívny
$(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$	$(p \vee q) \wedge r \equiv (p \wedge r) \vee (q \wedge r)$	distributívny
$\neg(p \wedge q) \equiv (\neg p) \vee (\neg q)$	$\neg(p \vee q) \equiv (\neg p) \wedge (\neg q)$	de Morganov
$p \wedge p \equiv p$	$p \vee p \equiv p$	idempotentnosti
$p \wedge (\neg p) \equiv 0$	$p \vee (\neg p) \equiv 1$	komplementárnosti
$\neg(\neg p) \equiv p$	$\neg(\neg p) \equiv p$	involúcie
$p \wedge (p \vee q) \equiv p$	$p \vee (p \wedge q) \equiv p$	absorpcie
$p \wedge 1 \equiv p$	$p \vee 0 \equiv p$	identity
$p \wedge 0 \equiv 0$	$p \vee 1 \equiv 1$	jednotkového sčitovania a nulového násobenia

Tab. 13: Zákony Boolovej algebry

Tab. 13 zobrazuje vlastnosti, ktoré platia v Boolovej algebre. Pomocou týchto zákonov dokážeme mnohé Boolovské funkcie zapísať jednoduchším spôsobom. [6]

4 ŁUKASIEWICZOVA LOGIKA

V klasickej dvojhodnotovej logike sú výroky buď len pravdivé alebo len nepravdivé. Má to ale svoje obmedzenia, pretože sa nevie ohodnotiť nejaká náhodná budúca udalosť. Preto v roku 1920 vykonal odklon od klasickej dvojhodnotovej logiky poľský filozof a logik Jan Łukasiewicz. K pravdivostným hodnotám 0 a 1 pridal ešte tretiu pravdivostnú hodnotu, ktorá dostala označenie ½ a interpretovala pojem „neviem“. Łukasiewiczova logika teda obsahuje trojhodnotovú množinu $\{0,½,1\}$. Takto môže táto logika pokryť väčší okruh tvrdení a pracovať s nimi. Nevýhodou však je, že v trojhodnotovej logike neplatia niektoré pravidlá z klasickej výrokovej logiky. [4][9]

Príklad 14: Príklady výrokov: „Zajtra bude pekné počasie.“
„Vo vesmíre sú ďalšie planéty ako naša Zem, kde existuje život.“

Príklad 15: Príklady neplatných pravidiel z výrokovej logiky: $p \wedge (\neg p) \neq 0$

$$p \vee (\neg p) \neq 1$$

4.1 Logické spojky v 3-hodnotovej logike

4.1.1 Negácia

Negovaný výrok je v Łukasiewiczovej logike interpretovaný tak isto ako v klasickej logike, len s tým rozdielom, že obsahuje hodnotu ½ (neviem), ktorého negáciou je taktiež hodnota ½. Výsledok negácie je zobrazený v tabuľke Tab. 14. [9]

Funkčné vyjadrenie logickej spojky negácie: $e(\neg p) = 1 - e(p)$. [4]

p	$\neg p$
0	1
½	½
1	0

Tab. 14: Negácia

4.1.2 Konjunkcia

Zložený výrok spojený pomocou konjunkcie je v Łukasiewiczovej logike pravdivý len vtedy, ak sú pravdivé obidva výroky *p*, *q*. Ak je aspoň jeden výraz nepravdivý, tak aj výsledný výrok je nepravdivý. V ostatných prípadoch má výsledný výraz pravdivostnú hodnotu „neviem“. Výsledok konjunkcie je zobrazený v tabuľke Tab. 15. [9]

Funkčné vyjadrenie logickej spojky konjunkcie: $e(p \wedge q) = \min\{e(p), e(q)\}$. [4]

p	q	$p \wedge q$
0	0	0
0	½	0
0	1	0
½	0	0
½	½	½
½	1	½
1	0	0
1	½	½
1	1	1

Tab. 15: Konjunkcia

4.1.3 Disjunkcia

Zložený výrok spojený pomocou disjunkcie je v Łukasiewiczovej logike nepravdivý len vtedy, ak sú nepravdivé obidva výroky *p*, *q*. Ak je aspoň jeden výraz pravdivý, tak aj výsledný výrok je pravdivý. V ostatných prípadoch má výsledný výraz pravdivostnú hodnotu „neviem“. Výsledok disjunkcie je zobrazený v tabuľke Tab. 16. [9]

Funkčné vyjadrenie logickej spojky disjunkcie: $e(p \vee q) = \max\{e(p), e(q)\}$. [4]

p	q	$p \vee q$
0	0	0
0	½	½
0	1	1
½	0	½
½	½	½
½	1	1
1	0	1
1	½	1
1	1	1

Tab. 16: Disjunkcia

4.1.4 Implikácia

Zložený výrok spojený pomocou implikácie je v Łukasiewiczovej logike tak ako v klasickej logike nepravdivý len vtedy, ak je prvý výrok pravdivý a druhý nepravdivý. Ak je hodnota prvého výroku ½ a druhého 0 alebo prvého 1 a druhého ½, je v týchto prípadoch pravdivostná hodnota implikácie ½, čiže „neviem“. V ostatných prípadoch je zložený výraz pravdivý. Výsledok implikácie je zobrazený v tabuľke Tab. 17. [9]

Funkčné vyjadrenie logickej spojky implikácie: $e(p \Rightarrow q) = \min\{1, 1 - e(p) + e(q)\}$. [4]

p	q	$p \Rightarrow q$
0	0	1
0	½	1
0	1	1
½	0	½
½	½	1
½	1	1
1	0	0
1	½	½
1	1	1

Tab. 17: Implikácia

4.2 Aplikácia Łukasiewiczovej logiky

Implementáciu tejto trojhodnotovej logiky je možné nájsť v dopytovacom jazyku SQL, kde je využitá pre zavedenie tretej logickej hodnoty *NULL*. Názorné ukážky implementácie sú podrobnejšie rozobrané v kapitole 10 Logické operátory v SQL.

5 FUZZY LOGIKA

Fuzzy logika je funkcionálna logika, v ktorej sa logické výroky ohodnocujú stupňom príslušnosti. Pravdivostné hodnoty sú rozložené v intervale $\langle 0{,}1 \rangle$, kde sú zahrnuté aj krajné hodnoty. Vo fuzzy logike tak môžeme vyjadriť pojmy ako „trochu“, „dosť“ atď., čiže sa môže vyjadriť čiastočná príslušnosť k množine. Vo fuzzy logike neplatí zákon vylúčenia tretieho a taktiež je viac možností pre výrokové spojky. Niekedy sa fuzzy logika zamieňa s pravdepodobnosťou, ale nie je tomu tak, pretože pravdepodobnosť vyjadruje neznámy výsledok ostrého javu. Naopak fuzzy logika vyjadruje známy stav neostrého javu. [10] [11]

Príklad 16:	Pravdepodobnosť:	Padne na kocke číslo 2?
	Ostrý jav:	Možnosti 1, 2, 3, 4, 5, 6.
	Stupeň pravdepodobnosti:	Vyjadruje s akou šancou padne číslo 2.
	Fuzzy logika:	Je z polovice odčerpaná nádrž prázdna?
	Neostrý jav:	Do úvahy sa berie celý rozsah nádrže.
	Stupeň pravdivosti:	Vyjadruje s ako pravdivosťou prislúcha stav k danej vlastnosti.

5.1 Fuzzy logické spojky

Výrokové spojky odpovedajú operáciám v intervale $\langle 0{,}1 \rangle$ pomocou pravdivostných funkcií, ktoré zovšeobecňujú pravdivostné tabuľky. [11]

5.1.1 Fuzzy negácia

Fuzzy negácia je unárna operácia, ktorá vyhovuje týmto podmienkam:

- $\neg(\neg p) \equiv p$,
- $e(p) \leq e(q) \Rightarrow e(\neg p) \geq e(\neg q)$.

Štandardná fuzzy negácia: $e(\neg p) = 1 - e(p)$. [4] [11]

5.1.2 Fuzzy konjunkcia

Fuzzy konjunkcia je binárna operácia, ktorá vyhovuje týmto podmienkam:

- komutatívnosť $p \wedge q \equiv q \wedge p$,
- asociatívnosť $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$,
- okrajová podmienka $p \wedge 1 \equiv p$,
- $e(q) \leq e(r) \Rightarrow e(p \wedge q) \leq e(p \wedge r)$.

Štandardná fuzzy konjunkcia: $e(p \wedge q) = \min\{e(p), e(q)\}$.

Súčinová konjunkcia: $e(p \wedge q) = e(p) \cdot e(q)$. [4] [11]

5.1.3 Fuzzy disjunkcia

Fuzzy disjunkcia je binárna operácia, ktorá vyhovuje týmto podmienkam:

- komutatívnosť $p \vee q \equiv q \vee p$,
- asociatívnosť $p \vee (q \vee r) \equiv (p \vee q) \vee r$,
- okrajová podmienka $p \vee 0 \equiv p$,
- $e(q) \leq e(r) \Rightarrow e(p \vee q) \leq e(p \vee r)$.

Štandardná fuzzy disjunkcia: $e(p \vee q) = \max\{e(p), e(q)\}$.

Súčinová disjunkcia: $e(p \vee q) = e(p) + e(q) - e(p) \cdot e(q)$. [4] [11]

5.1.4 Fuzzy implikácia

Fuzzy implikácia je binárna operácia, ktorá vyhovuje týmto okrajovým podmienkam:

- $e(p \Rightarrow q) = \begin{cases} 1, \ ak \ e(p) = 0 \ alebo \ e(q) = 1 \\ 0, \ ak \ e(p) = 1 \ a \ e(q) = 0 \end{cases}$.

Fuzzy implikácia: $e(p \Rightarrow q) = \min\{1, 1 - e(p) + e(q)\}$. [4]

5.2 Aplikácia fuzzy logiky

Fuzzy riadenie a regulácia je hlavná oblasť použitia fuzzy logiky, v ktorej sa využívajú dva spôsoby:

Priame fuzzy riadenie – predstavuje riadenie, v ktorom fuzzy regulátor prijíma hodnoty z riadeného systému, ktoré následne vyhodnocuje a na základe ich výsledku zasiela optimálne riadiace zásahy do systému

Nepriame fuzzy riadenie – predstavuje riadenie, v ktorom fuzzy regulátor prijíma hodnoty z riadeného systému, ktoré následne vyhodnocuje a na základe ich výsledku prepína medzi viacerými lineárnymi regulátormi, ktoré vykonávajú riadiace zásahy do systému

Fuzzy regulátor sa skladá z:

fuzzifikácie – premieňa vstupné hodnoty na fuzzy hodnoty,

tabuľky pravidiel – regulačný algoritmus systému,

defuzzifikácie – vyrába podľa vstupu a tabuľky pravidiel riadiace zásahy. [12]

Využitie fuzzy riadenia a regulácie môžeme nájsť v automobilovom priemysle, či už v systéme ABS, ktorý zabraňuje zablokovaniu kolies pri intenzívnom brzdení alebo v automatickej prevodovke, ktorá podľa charakteru jazdy určuje, pri akých otáčkach bude preraďovať rýchlostné stupne. [13][14]

Ďalšie použitie fuzzy systému môžeme nájsť v práčkach a umývačkách, kde fuzzy systém rozpozná, aké množstvo bielizne alebo riadu je vložené a podľa nečistoty vo vode určí mieru ich znečistenia, a tým optimalizuje dĺžku a intenzitu prania, prípadne umývania.[15]

Taktiež fuzzy systémy nájdeme v klimatizáciách, ktoré pomocou meranej teploty určujú vhodnú teplotu pre chladenie vzduchu alebo vo fotoaparátoch pre automatické zaostrovanie.[16]

6 LOGICKÉ OBVODY

Logický obvod je systém realizujúci prostredníctvom logických signálov logickú funkciu.

V logických obvodoch sa používajú skokovo premenné signály, ktoré môžu nadobudnúť dve hodnoty. Tieto dve hodnoty sa označujú ako logická jednotka a logická nula (log. 1, log. 0).

Signály môžu byť realizované rôznymi spôsobmi, medzi ktoré patria:

- logická interpretácia 1, 0,
- pravdivostná interpretácia, výrok pravdivý (1), nepravdivý (0),
- interpretácia vyjadrujúca aktívny (1) a neaktívny (0) stav určitej riadiacej veličiny,
- kontaktová reprezentácia, zopnuté (1), rozopnuté (0).

Realizácia logických funkcií predstavuje zostavovanie schém týchto obvodov, kde sa pomocou zadanej logickej funkcie vytvorí zo vstupných premenných výsledná výstupná premenná. Pojem logická funkcia je bližšie vysvetlený v kapitole 3 Boolova algebra.

Medzi najdôležitejšie logické funkcie v logických obvodoch patria:

- $y = a \cdot b$ logický súčin, konjunkcia, *And*,
- $y = \overline{a \cdot b}$ Shefferova funkcia, *Nand*,
- $y = a + b$ logický súčet, disjunkcia, *Or*,
- $y = \overline{a + b}$ Peircova funkcia, *Nor*,
- $y = a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b$ exkluzívny súčet, *Xor*,
- $y = \overline{a \oplus b} = a \cdot b + \overline{a} \cdot \overline{b}$ negácia exkluzívneho súčtu, *Xnor*. [17]

Bližší popis využitia logických funkcií v logických obvodoch je popísaný v praktickej časti bakalárskej práce v kapitole 9 Logické operátory v logických obvodoch, kde sú taktiež zobrazené ich praktické ukážky.

7 DATABÁZOVÉ SYSTÉMY

Databázový systém je informačný systém, ktorého úlohou je využitím programového vybavenia umožňovať prístup k súboru navzájom súvisiacich údajov a narábať s nimi. Je zostavený z týchto dvoch častí:

- databáza,
- systém riadenia bázy údajov.

Pod pojmom databáza rozumieme súbor údajov, údajových tabuliek a iných objektov, ktorých usporiadanie je také, aby podporovalo vykonávanie požiadaviek ako kombinovanie, vyhľadávanie a preusporiadavanie týchto údajov.

Systém riadenia bázy údajov je systém programov, ktoré vykonávajú operácie s údajmi. V súčasnosti najpoužívanejší systém riadenia bázy údajov je SQL jazyk. SQL jazyk je založený na relačnom modeli dát. Dáta sú používateľovi prezentované vo forme tabuľky. Tieto tabuľky a ich stĺpce majú svoje jednoznačne určené mená, s ktorými k nim možno pristupovať.

Základné syntaktické konštrukcie jazyka SQL sú:

- manipulácia dát
  - *SELECT* slúži pre výber dát z tabuľky,
  - *INSERT* umožňuje vloženie nových dát do tabuľky,
  - *DELETE* slúži na vymazanie dát z tabuľky,
  - *UPDATE* upravuje existujúce dáta v tabuľke.
- definícia dát
  - *CREATE* vytvára nové databázové objekty,
  - *ALTER* upravuje databázové objekty,
  - *DROP* ruší databázové objekty. [18][19]

Logické operátory sú používané v príkazoch pre manipuláciu dát, kde slúžia na presnú špecifikáciu podmnožiny údajov, ktoré majú byť vybrané zo zvolených tabuliek. Praktické ukážky ich zápisu sú bližšie popísane v kapitole 10 Logické operátory v SQL.

II. PRAKTICKÁ ČASŤ

8 LOGICKÉ OPERÁTORY VO WOLFRAM MATHEMATICA

Software Wolfram Mathematica zahŕňa vo svojom programovom vybavení funkcie a operátory matematickej logiky. Logické operátory softwaru Mathematica sú prevažne používané pri riešení úloh výrokovej logiky, ale taktiež pri určovaní definičných oborov úloh, zisťovaní väzieb medzi objektmi, určovaní veľkosti plochy a v neposlednom rade taktiež hľadaní rôznych extrémov funkcií na danom ohraničenom priestore. Boolove funkcie programu sa môžu využiť pri úlohách pre zostavovanie logických obvodov, pre minimalizovanie logických funkcií alebo ich prepis do nami žiadaného tvaru. V nasledujúcej kapitole sú popísané tieto operátory a funkcie, ich zápis a nakoniec ich využitie pri riešení úloh z matematiky.

8.1 Logické operátory a kvantifikátory

Všetky logické operátory môžu byť zapísané pomocou funkcie, ale taktiež aj pomocou špeciálne vyhradených symbolov. Význam pri tom zostáva nezmenený. Pravdivostné hodnoty sú vyjadrené pomocou výrazov *True* a *False*, ktoré znamenajú pravdivý, resp. nepravdivý výrok.

Not[p] – operátor negácie, ekvivalentný zápis $!p$ alebo $\neg p$.

And[p,q,...] – operátor konjunkcie, ekvivalentný zápis pre dva výroky p && q alebo $p \wedge q$.

Or[p,q,...] – operátor disjunkcie, ekvivalentný zápis pre dva výroky $p \parallel q$ alebo $p \vee q$.

Implies[p,q,...] – operátor implikácie, ekvivalentný zápis pre dva výroky $p \Rightarrow q$.

Equivalent[p,q,...] – operátor ekvivalencie, ekvivalentný zápis pre dva výroky $p \Leftrightarrow q$.

Xor[p,q,...] – operátor exkluzívneho súčtu, ekvivalentný zápis pre dva výroky $p \veebar q$.

Nand[p,q,...] – operátor Shefferovej funkcie, ekvivalentný zápis pre dva výroky $p \barwedge q$.

Nor[p,q,...] – operátor Piercovej funkcie, ekvivalentný zápis pre dva výroky $p \overline{\vee} q$.

Xnor[p,q,...] – operátor negácie exk. súčtu.

ForAll[ ] – reprezentuje všeobecný kvantifikátor.

Exists[ ] – reprezentuje existenčný kvantifikátor.

Príklad 17: Použitie logických operátorov:

In[1]:= $p = True$;

$q = False$;

p && q

$p \parallel q$

Nor[p, q]

Out[1]:= *False*

Out[2]:= *True*

Out[3]:= *False*

Použitie kvantifikátorov:

In[1]:= ForAll[x, $x^2 + x > -1$]

Out[1]:= $\forall_x \; x^2 + x > -1$

8.2 Boolove funkcie

Boolove funkcie softwaru Mathematica sa používajú prevažne na urýchlenie výpočtov a zjednodušovanie logických funkcií, ktoré sú využívané v logických obvodoch. Medzi základné Boolove funkcie patria:

Boole[ ] – funkcia, ktorá zobrazuje pravdivostné hodnoty do množiny $\{0,1\}$.

BooleanConvert[ ] – funkcia, ktorá upravuje Boolove funkcie do iných špeciálnych tvarov Boolových funkcií. Medzi tieto tvary patrí disjunktívny normálny tvar, skratka *DNF*, konjunktívny normálny tvar, *CNF*, ale taktiež tvar, v ktorom je logická funkcia zapísaná len pomocou spojok *Nand*, *Nor*, *And*, atď.

BooleanMinimize[ ] – nájde minimalizovaný tvar zadanej funkcie v disjunktívnom normálnom tvare.

BooleanFunction[ ] – vráti Boolovu funkciu po zadaní hodnôt z pravdivostnej tabuľky.

SatisfiabilityInstances[ ] – zobrazí hodnoty vstupných výrokových premenných, pri ktorých má Boolova funkcia pravdivé pravdivostné ohodnotenie.

Príklad 18: Použitie Boolových funkcií:

In[1]:= Boole[{*True*, *False*}]

Out[1]:= {1, 0}

In[2]:= BooleanMinimize[$c \wedge (a \veebar b) \vee a \wedge b$]

Out[2]:= (a && b) || (a && c) || (b && c)

In[3]:= BooleanConvert[$a \Rightarrow b$, "*NAND*"]

BooleanConvert[$a \Rightarrow b$, "*DNF*"]

BooleanConvert[$a \Rightarrow b$, "*NOR*"]

Out[3]:= $a \barwedge\, !b$

Out[4]:= $!a \,\|\, b$

Out[5]:= $!(!a \,\bar{\vee}\, b)$

In[6]:= BooleanFunction[{{*False*, *False*} → *False*, {*False*, *True*} → *False*, {*True*, *False*} → *True*, {*True*, *True*} → *False*}, {*a*, *b*}]

Out[6]:= a && !b

8.3 Ďalšie funkcie

Medzi ďalšie príkazy, ktoré sa používajú prevažne pri kvantifikovaných výrokoch, sa môžu zaradiť tieto funkcie:

FullSimplify[ ] – pretransformuje výraz do najjednoduchšieho možného tvaru.

Resolve[ ] – táto funkcia rieši výraz tak, že ho vyjadrí vo forme, v ktorej eliminuje všeobecné a existenčné kvantifikátory.

8.4 Aplikácia v matematických príkladoch

Logické operátory sa využívajú v rôznych typoch príkazov od podmienok *If[ ]* až po hľadanie extrémov funkcií. Medzi príkazy, kde sa najčastejšie tieto operátory vyskytujú, patria:

RegionPlot[ ] – funkcia vykreslí oblasť, v ktorej je výraz zapísaný pomocou logických kombinácií rovností a nerovností pravdivý.

Integrate[ ] – spočíta určitý integrál cez oblasť, v ktorej je výraz zapísaný pomocou logických kombinácií rovností a nerovností pravdivý.

FindMaximum[ ] – nájde maximum zadanej funkcie v oblasti, ktorá je ohraničená logickou kombináciou obmedzení.

FindMinimum[ ] – nájde minimum zadanej funkcie v oblasti, ktorá je ohraničená logickou kombináciou obmedzení.

8.5 Riešené ukážkové príklady

Príklady 19 a 20 znázorňujú použitie logických operátorov pri určovaní definičného oboru funkcie a taktiež ich použitie pri určovaní polohy zadaného bodu alebo oblasti voči definičnému oboru funkcie.

Príklad 19: Určte a vykreslite definičný obor funkcie $f(x,y) = \sqrt{x+y} + ArcSin[3+x]$. Je bod $A = [-2, 1.5]$ vnútorným bodom $D(f)$?

Riešenie:

Ako prvé vykonáme to, že pomocou funkcie *Reduce[ ]* určíme definičný obor funkcie.

In[1]:= Reduce[$-\infty < \sqrt{x+y} + ArcSin[3+x] < \infty$, {*x*,*y*}, Reals]

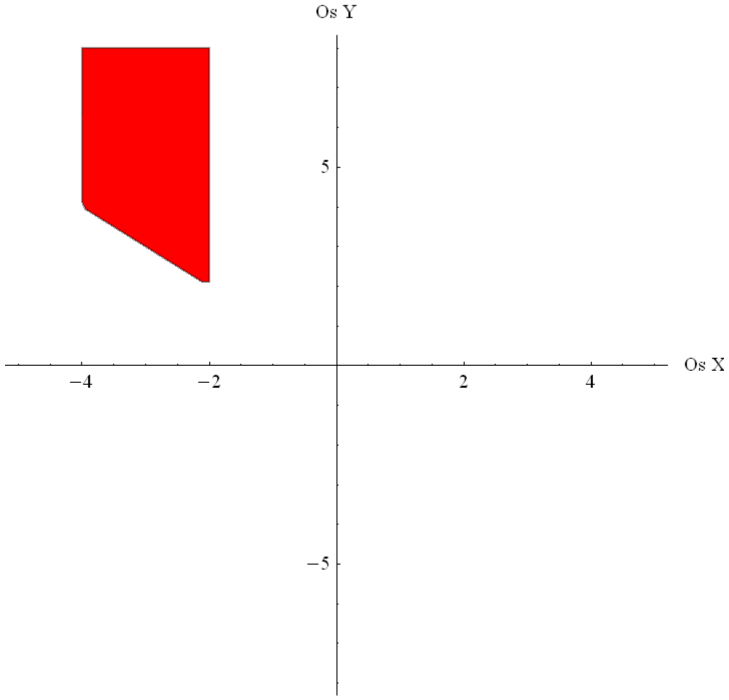
Out[1]:= $-4 \le x \le -2 \,\&\&\, y \ge -x$

Pomocou nájdeného definičného oboru funkcie môžeme príkazom *RegionPlot[ ]* vykresliť definičný obor funkcie $D(f)$.

In[2]:= *df* = RegionPlot[$-4 \le x \le -2 \,\&\&\, y \ge -x$, {*x*, -5, 5}, {*y*, -8, 8}, Axes → True, Frame → False, AxesLabel → {"Os X", "Os Y"}, PlotStyle → Red]

Ako je vidieť, do funkcie *RegionPlot[ ]* zapíšeme oblasť pomocou logických kombinácií nerovností, tzn. že využijeme logické operátory pre podmienky platnosti funkcie $f(x,y)$.

Out[2]:=



Obr. 1: Vykreslený definičný obor funkcie $f(x, y)$

Nakoniec zistíme či bod $A = [-2, 1.5]$ je vnútorným bodom $D(f)$ a zakreslíme ho do spoločného grafu. Spôsob, ako to zistiť je, že pomocou operátora /. priradíme hodnoty premenným *x* a *y*. Ak bude výsledok *True*, znamená to, že bod je vnútorným bodom $D(f)$, ak *False*, tak nie je.

In[3]:= $-4 \le x \le -2 \,\&\&\, y \ge -x$ /. {$x \to$ -2, $y \to$ 1.5}

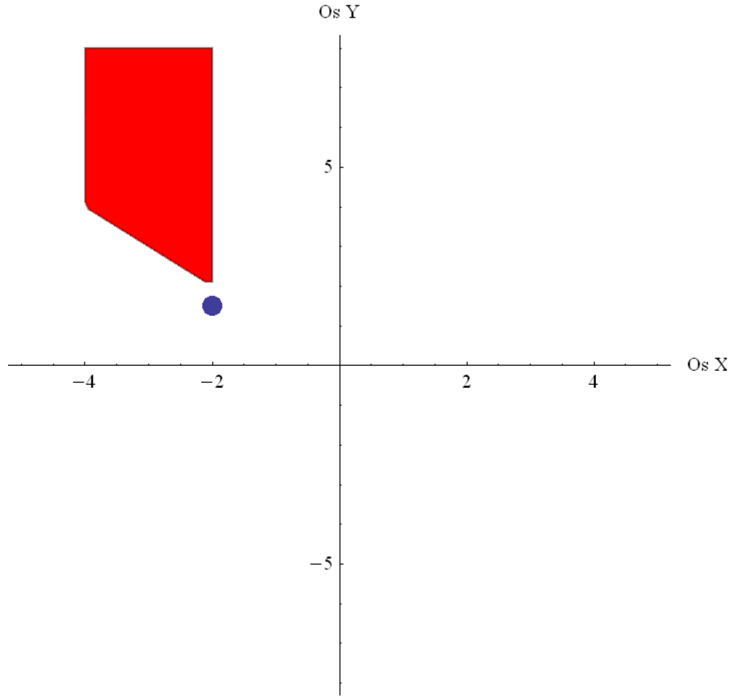
Out[3]:= *False*

Bod $A = [-2, 1.5]$ nie je vnútorným bodom $D(f)$. Bod zakreslíme do zobrazeného $D(f)$ tak, že pomocou funkcie *Show[ ]* spojíme vykreslenie $D(f)$ a bodu *A* do spoločného grafu.

In[4]:= *pt* = ListPlot[{{-2, 1.5}}, PlotStyle → PointSize[0.03]];

In[5]:= Show[*df*, *pt*]

Out[5]:=



Obr. 2: Vykreslený $D(f)$ funkcie $f(x,y)$ s bodom A

Z Obr. 2 je vidieť, že bod *A* naozaj nie je vnútorným bodom definičného oboru funkcie.

Príklad 20: Určte definičný obor funkcie $f(x,y) = Log\left[-\left(x\cdot\left(x+\frac{y}{2}+1\right)+y\cdot\left(\frac{x}{2}+y+1\right)\right)\right]$. Nachádza sa oblasť tvorená obmedzením $(x+0.3)^2+(y+0.3)^2 \le 0.15$ vo vnútri $D(f)$?

Riešenie:

Postup je podobný ako v predchádzajúcom príklade. Najprv zistíme definičný obor funkcie, uložíme do pomocnej premennej *dff* a zadanú oblasť do premennej *zo*.

In[1]:= $dff = \text{Reduce}[-\infty < Log\left[-\left(x\cdot\left(x+\frac{y}{2}+1\right)+y\cdot\left(\frac{x}{2}+y+1\right)\right)\right] < \infty, \{x,y\}, \text{Reals}]$

Out[1]:= $-1 < x < \frac{1}{3} \&\& \frac{1}{2}(-1-x)-\frac{1}{2}\sqrt{1-2x-3x^2} < y < \frac{1}{2}(-1-x)+\frac{1}{2}\sqrt{1-2x-3x^2}$

In[2]:= $zo = (x+0.3)^2+(y+0.3)^2 \le 0.15;$

Pomocou všeobecného kvantifikátora a funkcie *FullSimplify[ ]* zistíme či sa zadaná oblasť nachádza vo vnútri definičného oboru funkcie. Ak bude výsledok *True*, znamená to, že sa oblasť nachádza vo vnútri $D(f)$, ak *False*, tak nie.

In[3]:= FullSimplify[ForAll[{*x*, *y*}, *zo* ⇒ *dff*]]

Out[3]:= *True*

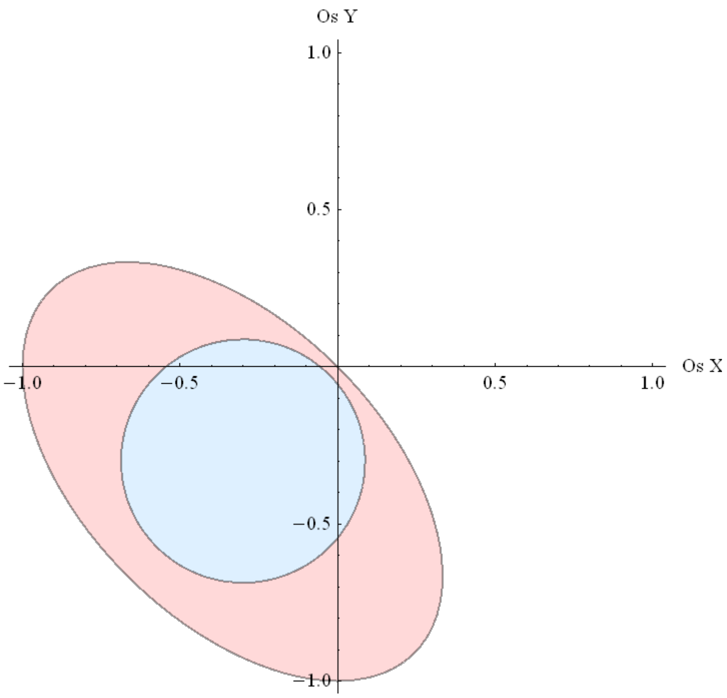
Z výsledku je zrejmé, že zadaná oblasť sa nachádza vo vnútri $D(f)$. Nakoniec zakreslíme nájdený $D(f)$ a zadanú oblasť do spoločného grafu pre overenie správnosti výsledku.

In[4]:= *plotdff* = RegionPlot[*dff*, {*x*, -1, 1}, {*y*, -1, 1}, Axes → True, Frame → False, AxesLabel → {"Os X", "Os Y"}, PlotStyle → LightRed];

In[5]:= *plotzo* = RegionPlot[*zo*, {*x*, -1, 1}, {*y*, -1, 1}, Axes → True, Frame → False, AxesLabel → {"Os X", "Os Y"}, PlotStyle → LightBlue];

In[6]:= Show[*plotdff*, *plotzo*]

Out[6]:=



Obr. 3: Vykreslený $D(f)$ funkcie $f(x, y)$ so zadanou oblasťou

Z Obr. 3 je vidieť, že zadaná oblasť sa nachádza vo vnútri $D(f)$.

Príklady 21 a 22 znázorňujú využitie logického operátora *And[ ]* pri riešení optimalizačných úloh klasického a neklasického viazaného extrému funkcie.

Príklad 21: Zistite maximum funkcie $F(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 \cdot x_3$ pri obmedzeniach $x_1 + x_2 = 2$ a $x_2 + x_3 = 2$.

Riešenie:

In[1]:= $\text{FindMaximum}\left[\{x_1 \cdot x_2 + x_2 \cdot x_3,\ And\left[(x_1 + x_2 = 2), (x_2 + x_3 = 2)\right]\}, \{x_1, x_2, x_3\}\right]$

Out[1]:= $\{2., \{x_1 \to 1.,\ x_2 \to 1.,\ x_3 \to 1.\}\}$

Z výsledku môžeme vidieť, že maximum funkcie $F(x_1, x_2, x_3)$ je v bode $[1, 1, 1]$ a jeho hodnota je 2.

Príklad 22: Zistite maximum funkcie $F(x_1, x_2) = 1.5 \cdot x_1 + 2 \cdot x_2$ pri obmedzeniach $x_1 + 2 \cdot x_2 \leq 36$ a $3 \cdot x_1 + x_2 \leq 24$.

Riešenie:

In[1]:= $\text{FindMaximum}\left[\{1.5 \cdot x_1 + 2 \cdot x_2,\ And\left[(x_1 + 2 \cdot x_2 \leq 36), (3 \cdot x_1 + x_2 \leq 24)\right]\}, \{x_1, x_2\}\right]$

Out[1]:= $\{37.2, \{x_1 \to 2.4,\ x_2 \to 16.8\}\}$

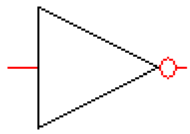
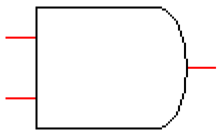
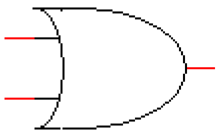
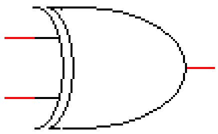
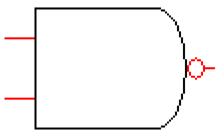
Z výsledku je zrejmé, že maximum funkcie $F(x_1, x_2)$ je v bode $[2.4, 16.8]$ o veľkosti 37.2.

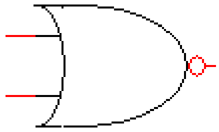
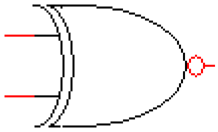
9 LOGICKÉ OPERÁTORY V LOGICKÝCH OBVODOCH

Tak ako bolo písané v teoretickej časti, logický obvod je systém realizujúci prostredníctvom logických signálov logickú funkciu. Logické obvody delíme na dva typy. Kombinačné logické obvody a sekvenčné logické obvody. Pri kombinačných obvodoch je stav výstupov jednoznačne daný stavom aktuálnych vstupov, pri sekvenčných záleží okrem aktuálneho stavu vstupov aj na minulom stave výstupu.

9.1 Logické hradlá

Logické kombinačné obvody sa delia na jednoduchšie a zložitejšie. Medzi jednoduchšie patria hradlá, ktoré vykonávajú základné logické operácie matematickej logiky ako sú negácia, konjunkcia, disjunkcia, atď.

Hradlo	**Logická funkcia**		**Schematická značka**
Not	Negácia	$y = \overline{x}$ $y = \neg x$	
And	Konjunkcia	$y = x \cdot y$ $y = x \wedge y$	
Or	Disjunkcia	$y = x + y$ $y = x \vee y$	
Xor	Exkluzívny súčet	$y = x \oplus y$ $y = \overline{x} \cdot y + x \cdot \overline{y}$ $y = (\neg x) \wedge y \vee x \wedge (\neg y)$	
Nand	Shefferova funkcia	$y = \overline{x \cdot y}$ $y = \neg(x \wedge y)$	

Nor	Piercova funkcia	$y = \overline{x + y}$ $y = \neg(x \vee y)$	
Xnor	Negácia exkluzívneho súčtu	$y = \overline{x \oplus y}$ $y = x \cdot y + \bar{x} \cdot \bar{y}$ $y = x \wedge y \vee (\neg x) \wedge (\neg y)$	

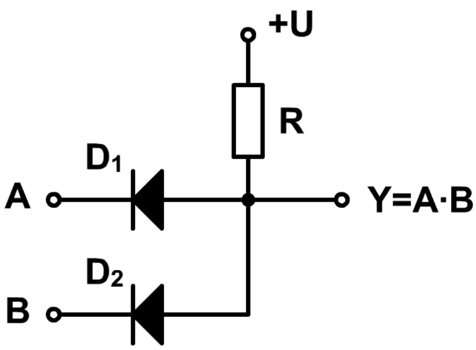
Tab. 18: Logické hradlá

9.2 Štruktúra logických hradiel

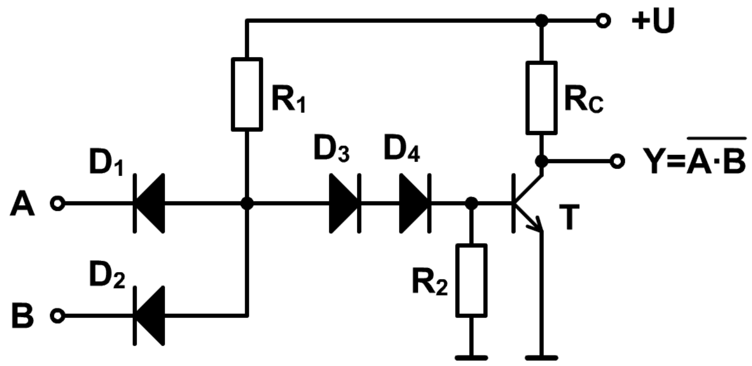
Logické hradlá môžeme realizovať rôznymi spôsobmi. [20] Medzi základné spôsoby realizácie týchto obvodov patria:

- Diódová logika (DL),
- Diódovo-tranzistorová logika (DTL),
- Odporovo-tranzistorová logika (RTL),
- Tranzistorovo-tranzistorová logika (TTL),
- CMOS logické obvody.

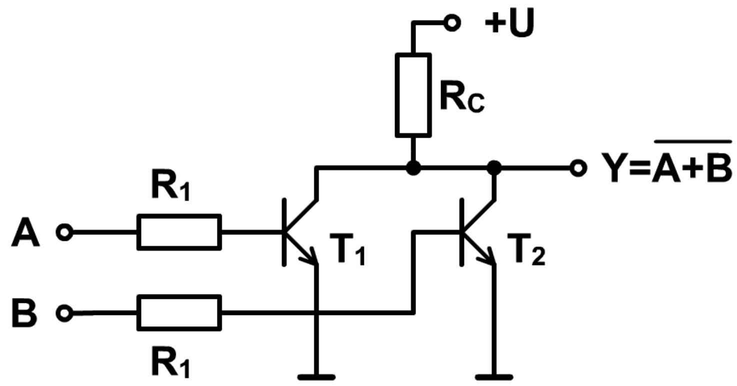
Nasledujúce obrázky zobrazujú elektronickú realizáciu vybraných logických hradiel v rôznych typoch zapojenia.



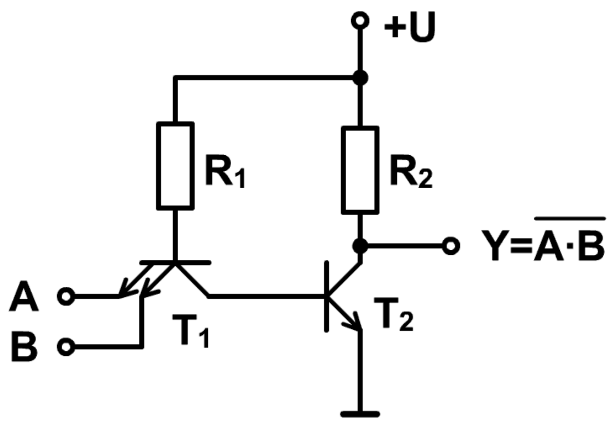
Obr. 4: Hradlo *And* v DL logike [20]



Obr. 5: Hradlo *Nand* v DTL logike [20]



Obr. 6: Hradlo *Nor* v RTL logike [20]



Obr. 7: Hradlo *Nand* v TTL logike [20]

9.3 Využitie hradiel v kombinačných obvodoch

Zložitejšie kombinačné obvody sú realizované pomocou hradiel a patria medzi ne obvody ako je multiplexor, demultiplexor, komparátor, kodér, dekodér a ďalšie. Nemenej významné sú taktiež kombinačné obvody ako je sčítačka, odčítačka, násobička, delička, tvoriace podsystémy aritmeticko-logickej jednotky ALU, ktorá patrí medzi základné časti procesora CPU.

Logické obvody sa vytvárajú pomocou pravdivostnej tabuľky, z ktorej sa vyjadrí logická funkcia. Následne sa zostrojí pomocou hradiel logický obvod, ktorý bude túto funkciu realizovať.

Príklad 23: Zostavte pomocou polovičnej a úplnej sčítačky 4-bitovú binárnu sčítačku, ktorá sčíta 2 binárne čísla. Polovičnú a úplnú sčítačku realizujte dvojvstupovými hradlami *Xor* a *Nand*.

Riešenie:

Ako prvé si zostavíme pravdivostné tabuľky pre polovičnú a úplnú sčítačku, kde A_n, B_n predstavujú n-té bity binárnych čísiel, S_n ich n-tý súčet a C_{n+1} prenos do vyššieho bitu. Tzn., že ak sčítavame nulté bity binárnych čísiel A_0 a B_0, tak C_1 tvorí prenos do prvého bitu, teda tento výstup privedieme na vstup do úplnej sčítačky, kde súčet prvého bitu tvorí súčet prvých bitov binárnych čísiel A_1, B_1 a prenosu C_1.

A_0	B_0	S_0	C_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tab. 19: Pravdivostná tabuľka polovičnej sčítačky

A_1	B_1	C_1	S_1	C_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tab. 20: Pravdivostná tabuľka úplnej sčítačky

Následne vytvoríme funkciu v disjunktnom normálnom tvare, tzn. že spíšeme súčet základných súčinov pre každú pravdivostnú tabuľku. Základné súčiny pre logickú funkciu S_0 vypíšeme tak, že vyberieme riadky, kde S_0 nadobúda hodnoty 1. Základné súčiny teda sú $\overline{A_0} \cdot B_0$ a $A_0 \cdot \overline{B_0}$.

Funkcia S_0 v disjunktnom normálnom tvare: $S_0 = \overline{A_0} \cdot B_0 + A_0 \cdot \overline{B_0}$

Takto spíšeme disjunktné normálne tvary aj pre funkcie C_1, S_1 a C_2.

Funkcia C_1: $C_1 = A_0 \cdot B_0$

Funkcia S_1: $S_1 = \overline{A_1} \cdot \overline{B_1} \cdot C_1 + \overline{A_1} \cdot B_1 \cdot \overline{C_1} + A_1 \cdot \overline{B_1} \cdot \overline{C_1} + A_1 \cdot B_1 \cdot C_1$

Funkcia C_2: $C_2 = \overline{A_1} \cdot B_1 \cdot C_1 + A_1 \cdot \overline{B_1} \cdot C_1 + A_1 \cdot B_1 \cdot \overline{C_1} + A_1 \cdot B_1 \cdot C_1$

Pomocou základných vzťahov Boolovej algebry a logických funkcií upravíme funkciu tak, aby sme využili pri realizácii funkcie len hradlá *Xor* a *Nand*.

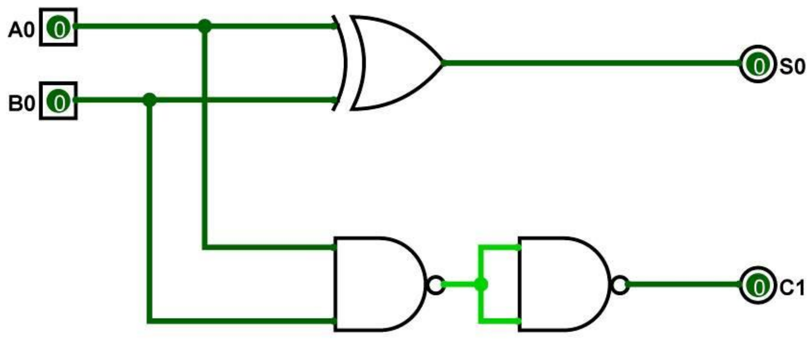
$$S_0 = \overline{A_0} \cdot B_0 + A_0 \cdot \overline{B_0}$$
$$S_0 = A_0 \oplus B_0$$

$$C_1 = A_0 \cdot B_0$$
$$C_1 = \overline{\overline{A_0 \cdot B_0}}$$

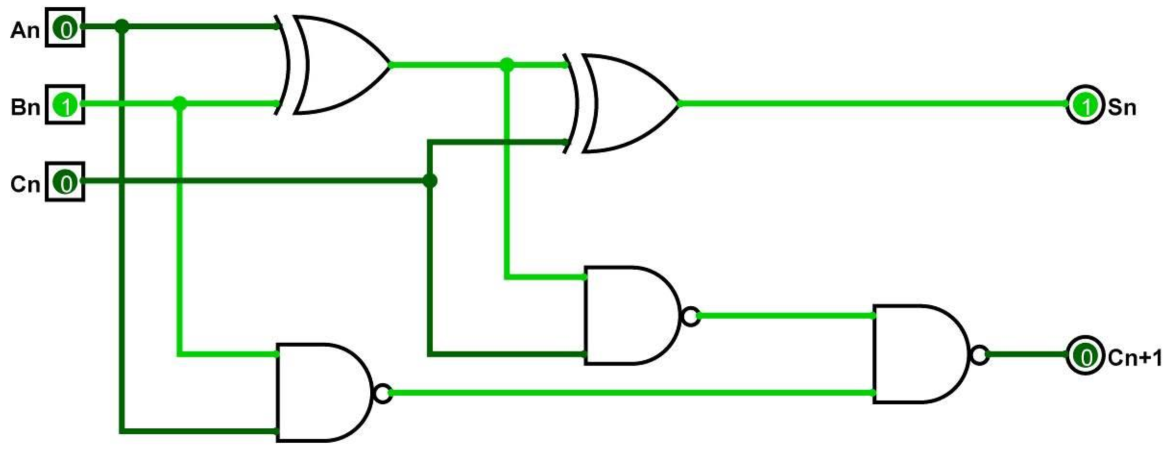
$$S_1 = \overline{A_1} \cdot \overline{B_1} \cdot C_1 + \overline{A_1} \cdot B_1 \cdot \overline{C_1} + A_1 \cdot \overline{B_1} \cdot \overline{C_1} + A_1 \cdot B_1 \cdot C_1$$
$$S_1 = \overline{A_1} \cdot \left(\overline{B_1} \cdot C_1 + B_1 \cdot \overline{C_1}\right) + A_1 \cdot \left(\overline{B_1} \cdot \overline{C_1} + B_1 \cdot C_1\right)$$
$$S_1 = \overline{A_1} \cdot \left(B_1 \oplus C_1\right) + A_1 \cdot \left(\overline{B_1 \oplus C_1}\right)$$
$$S_1 = A_1 \oplus \left(B_1 \oplus C_1\right)$$
$$S_1 = A_1 \oplus B_1 \oplus C_1$$

$$C_2 = \overline{A_1} \cdot B_1 \cdot C_1 + A_1 \cdot \overline{B_1} \cdot C_1 + A_1 \cdot B_1 \cdot \overline{C_1} + A_1 \cdot B_1 \cdot C_1$$
$$C_2 = C_1 \cdot \left(\overline{A_1} \cdot B_1 + A_1 \cdot \overline{B_1}\right) + A_1 \cdot B_1 \cdot \left(\overline{C_1} + C_1\right)$$
$$C_2 = C_1 \cdot \left(A_1 \oplus B_1\right) + A_1 \cdot B_1$$
$$C_2 = \overline{\overline{C_1 \cdot \left(A_1 \oplus B_1\right) + A_1 \cdot B_1}}$$
$$C_2 = \overline{\overline{C_1 \cdot \left(A_1 \oplus B_1\right)} \cdot \overline{A_1 \cdot B_1}}$$

Po úprave logických funkcií môžeme zostaviť výsledné logické schémy pre polovičnú a úplnú sčítačku.



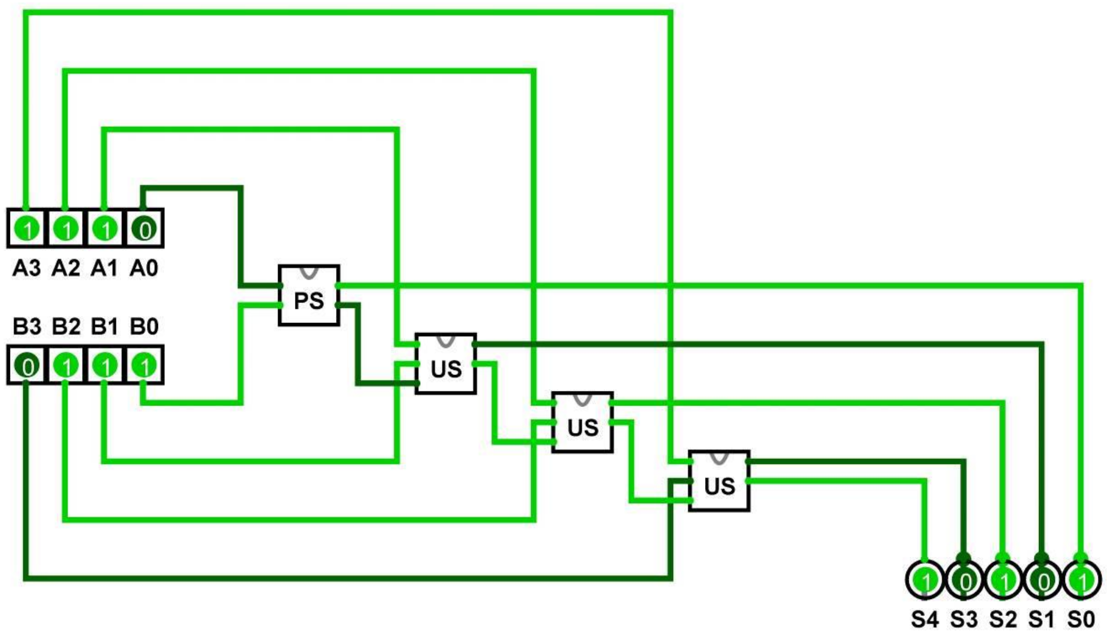
Obr. 8: Výsledná schéma pre polovičnú sčítačku



Obr. 9: Výsledná schéma pre úplnú sčítačku

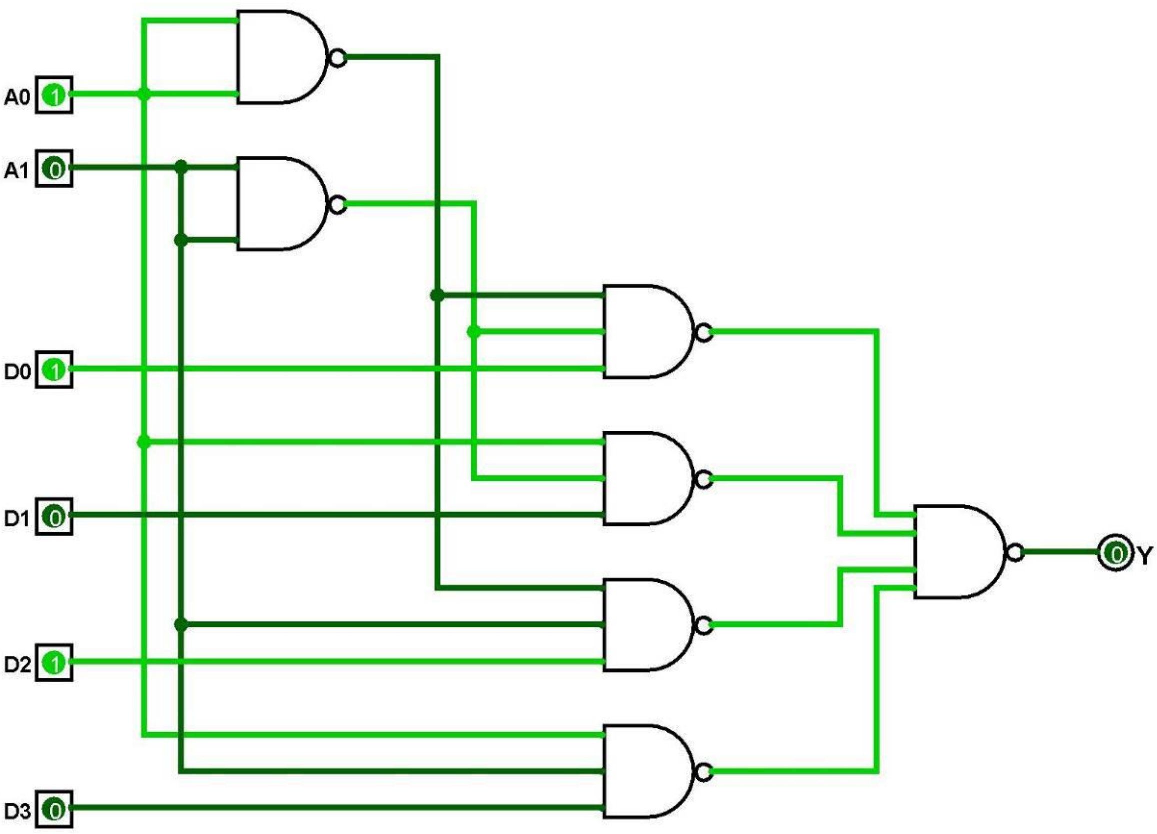
Tieto výsledné schémy sú zostrojené v softwari Logisim, ktorý taktiež ponúka každú schému reprezentovať samostatným funkčným blokom. Schému polovičnej sčítačky predstavuje *funkčný blok PS* s dvomi vstupmi a dvomi výstupmi. Schému úplnej sčítačky predstavuje *funkčný blok US* s tromi vstupmi a dvomi výstupmi. Nakoniec zostavíme 4-bitovú sčítačku, čiže zapojíme za sebou jednu polovičnú sčítačku a ďalej postupne úplné sčítačky tak, že prenos bitu z predchádzajúcej sčítačky zapojíme do ďalšej v poradí. Takto docielime vytvorenie 4-bitovej sčítačky.

Výsledné riešenie:



Obr. 10: Výsledná schéma pre 4-bitovú sčítačku

Príklad 24: Zostavte logickú funkciu v disjunktívnom normálnom tvare pre multiplexor s dvomi adresnými bitmi.



Obr. 11: Schéma pre multiplexor

Riešenie:

Pri zostavovaní logickej funkcie zo schémy postupujeme sprava doľava. Vidíme, že funkcia *Y* je zostavené z negácie 4 základných súčinov a to tak, že hradlo *Nand* má 4 vstupy.

$$Y = \overline{\_ \cdot \_ \cdot \_ \cdot \_}$$

Po prechode prvým *Nand* hradlom môžeme ďalej určiť, že každý zo 4 súčinov sa skladá z negácie 3 súčinov.

$$Y = \overline{\overline{\_ \cdot \_ \cdot \_} \cdot \overline{\_ \cdot \_ \cdot \_} \cdot \overline{\_ \cdot \_ \cdot \_} \cdot \overline{\_ \cdot \_ \cdot \_}}$$

Po tomto prechode sa dostávame k základným vstupným signálom, tzn., že do voľných políčok zapíšeme premenné týchto signálov.

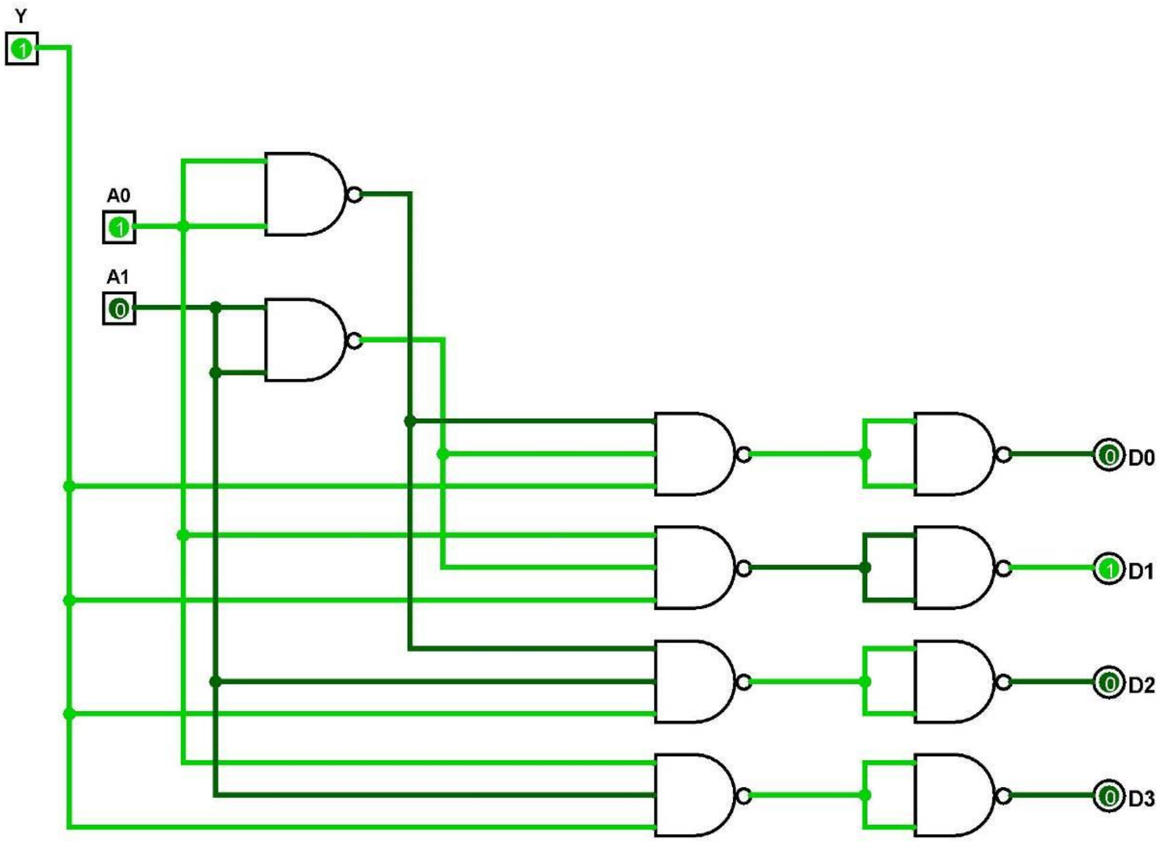
$$Y=\overline{\overline{\overline{A_0}\cdot\overline{A_1}\cdot D_0}\cdot\overline{A_0\cdot\overline{A_1}\cdot D_1}\cdot\overline{\overline{A_0}\cdot A_1\cdot D_2}\cdot\overline{A_0\cdot A_1\cdot D_3}}$$

Ako posledné už len upravíme pomocou de Morganových zákonov funkciu do disjunktívneho normálneho tvaru.

Výsledné riešenie:

$$Y=\overline{A_0}\cdot\overline{A_1}\cdot D_0+A_0\cdot\overline{A_1}\cdot D_1+\overline{A_0}\cdot A_1\cdot D_2+A_0\cdot A_1\cdot D_3$$

Riešenie príkladov pre ostatné logické kombinačné obvody sa vykonávajú týmito postupmi. Pre ďalšie znázornenie logických hradiel *Nand* v kombinačných obvodoch je znázornená schéma demultiplexoru v Obr. 12.



Obr. 12: Schéma zapojenia hradiel *Nand* pre demultiplexor

9.4 Využitie hradiel v sekvenčných obvodoch

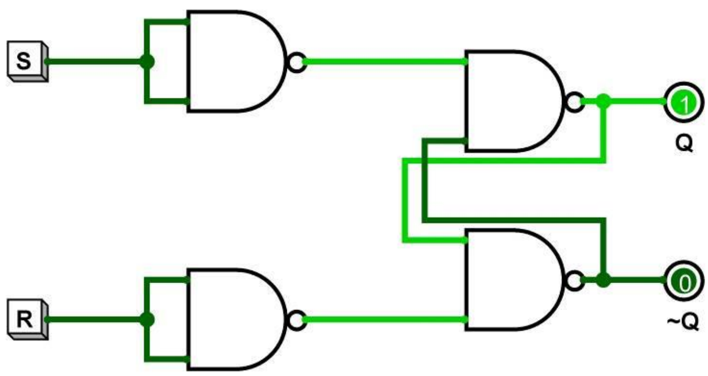
Medzi sekvenčné obvody patria preklápacie obvody, pomocou ktorých môžeme vytvoriť zložitejšie sekvenčné obvody ako sú počítadlo impulzov, registre, atď. Sekvenčné obvody sa skladajú z kombinačnej a pamäťovej časti. Kombinačná časť je štandardný kombinačný obvod a pamäťová časť je tvorená kombinačným obvodom so spätnou väzbou, ktorá privádza signál z výstupu naspäť na vstup.

Základnými sekvenčnými obvodmi sú RS obvod, D obvod, T obvod a JK obvod.

RS obvod – najzákladnejší sekvenčný obvod, tvorí základ pre ďalšie sekvenčné obvody ako sú D, T, JK. Tento obvod má 2 vstupy *R* (nuluj) a *S* (nastav), 2 výstupy, ktoré sú voči sebe negované a spätnú väzbu. Nevýhodou tohto preklápacieho obvodu je, že ak sa privedie zároveň na oba dva vstupy signál *log. 1*, dostane sa tento obvod do zakázaného stavu. Ak sa na oba vstupy privádza signál *log. 0*, obvod si pamätá predchádzajúci stav.

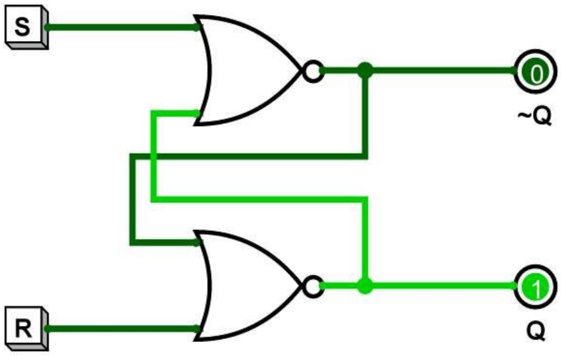
Pre zostavenie RS obvodu môžeme použiť dva typy zapojenia, t.j. pomocou hradiel *Nand* alebo pomocou hradiel *Nor*.

Pri zostavení RS obvodu hradlami *Nand* sú využité 4 tieto hradlá, kde prvé dve slúžia na negovanie vstupov a pri druhých dvoch privedieme k prvému hradlu výstup z druhého a naopak, teda zabezpečíme spätnú väzbu.



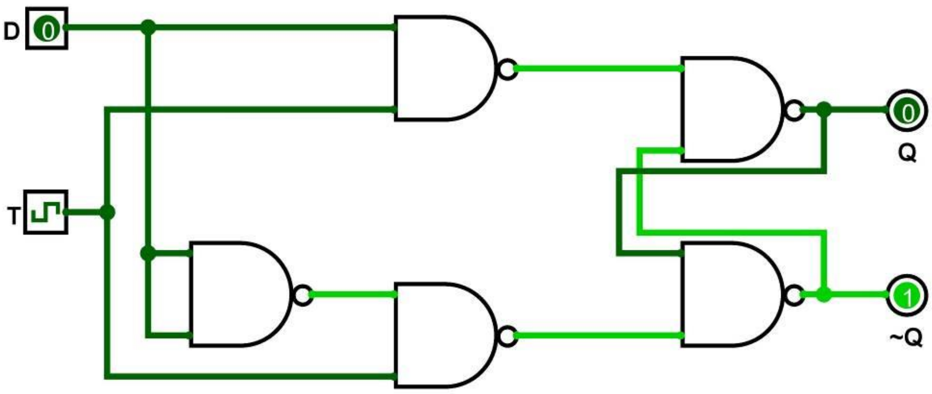
Obr. 13: Schéma RS preklápacieho obvodu - hradlá *Nand*

Pri zostavení RS obvodu hradlami *Nor* sú využité 2 tieto hradlá.



Obr. 14: Schéma RS preklápacieho obvodu - hradlá *Nor*

D preklápací obvod – predchádza zakázanému stavu, pretože je zapojený tak, že riadiaci signál *D* je len jeden, kde do prvého hradla *Nand* privádzame priamy signál a do druhého jeho negáciu. Týmto docielime, aby na týchto vstupoch bol vždy opačný signál.



Obr. 15: Schéma D preklápacieho obvodu

Ako ďalšie preklápacie obvody poznáme obvod T alebo JK. Takisto ako D obvod, aj tieto vychádzajú zo zapojenia RS preklápacieho obvodu, len ich devízou je, že na rozdiel od RS obvodu zabraňujú zakázanému stavu.

10 LOGICKÉ OPERÁTORY V SQL

Logické operátory sa v databázovom jazyku SQL využívajú v príkazy *SELECT*, ktorý vracia množinu záznamov z tabuliek, prípadne v príkazoch *DELETE, UPDATE*. Príkazy *SELECT, DELETE, UPDATE* používajú pomocný príkaz *WHERE*, práve v ktorom nájdeme uplatnenie logických operátorov. Tento pomocný príkaz slúži na zápis filtrovacích podmienok pre výber riadkov.

10.1 Prehľad logický operátorov

V jazyku SQL sa používajú nasledovné logické operátory:

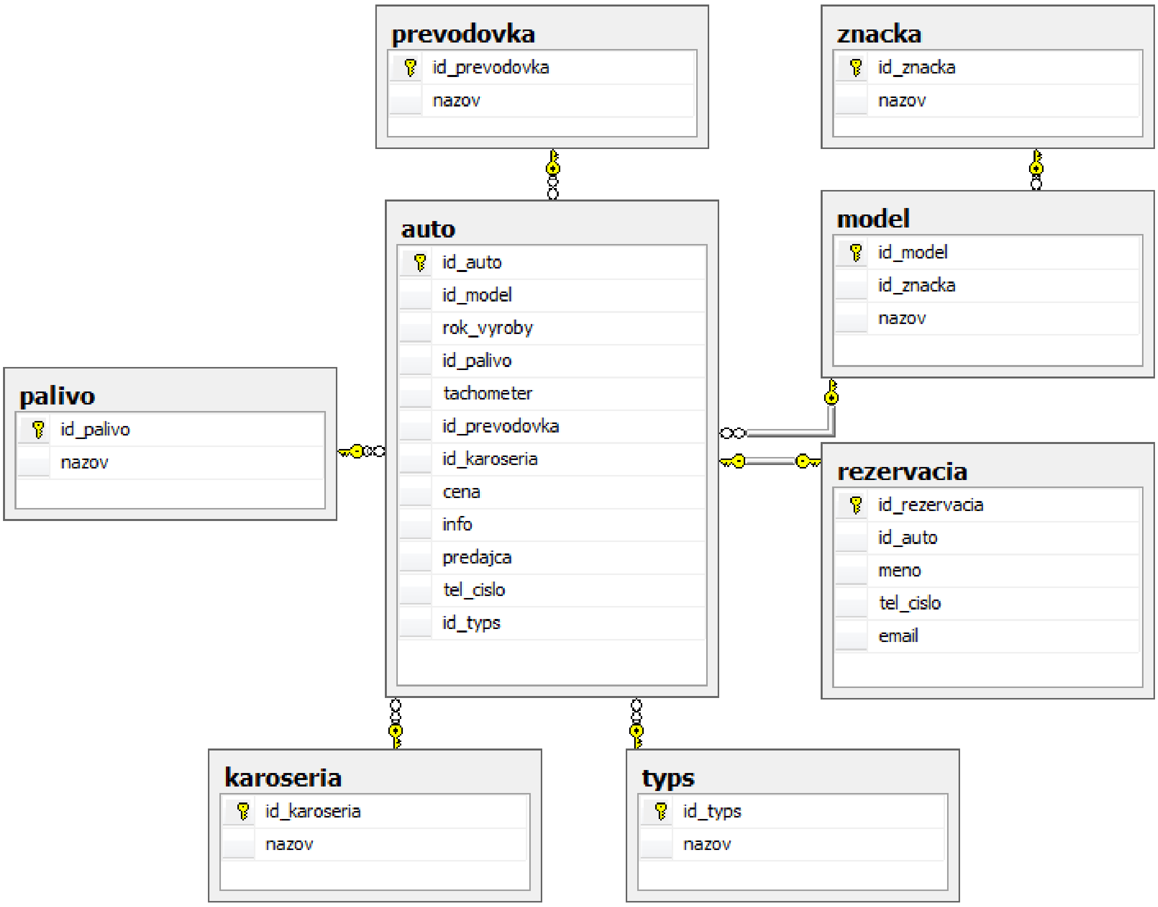
Operátor	Popis
Not	Obracia hodnotu každého logického operátora.
And	Pravda, pokiaľ sú obidve spájané podmienky pravdivé.
Or	Pravda, ak je aspoň jedna podmienka pravdivá.
Exists	Pravda, ak vrátený poddopyt obsahuje nejaké riadky.

Tab. 21: Prehľad logických operátorov v SQL

10.2 Využitie logických operátorov

Využitie logických operátorov ukážem na databázovom systéme *Autodatabase,* vytvorenom v softwari Microsoft SQL Server, ktorý sa nachádza v prílohe bakalárskej práce. Tento databázový systém simuluje databázu nových alebo ojazdených áut, ktoré si môžu zákazníci podľa zadaných kritérií prehliadať, prípadne auto zarezervovať. Využité logické operátory sú v príkazoch *SELECT* zaznačené zelenou farbou písma.

Databázový systém *Autodatabase* obsahuje 9 tabuliek. Hlavnými tabuľkami databázy sú tabuľka *Auto* a tabuľka *Rezervácia*. Pomocou vzťahu 1:N sú prepojené ostatné tabuľky *Model*, *Prevodovka*, *Palivo*, *Karoséria*, *Typs* s tabuľkou *Auto*, kde týmto prepojením určujeme, o aký model auta sa jedná a aké sú jeho základné parametre. Tabuľka *Rezervácia* a tabuľka *Auto* sú prepojené vzťahom 1:1, tzn., že auto môže byť zarezervované len jedným zákazníkom.



Obr. 16: Diagram databázového systému *Autodatabase*

Príklad 25: Zobrazenie aút, ktoré tankujú naftu, majú manuálnu prevodovku a ich cena je rovná alebo menšia ako 6000€.

```
select      znacka.nazov as 'Značka', model.nazov as 'Model',
            YEAR(rok_vyroby)  as  'Rok  výroby',  palivo.nazov  as  'Palivo',
            tachometer  as  'Najazdené',  prevodovka.nazov  as  'Prevodovka',
            cena as 'Cena'
from dbo.auto

inner join model on
      model.id_model = auto.id_model
inner join znacka on
      znacka.id_znacka = model.id_znacka
inner join palivo on
      palivo.id_palivo = auto.id_palivo
inner join prevodovka on
      prevodovka.id_prevodovka = auto.id_prevodovka

where (palivo.nazov='Diesel')
      AND (prevodovka.nazov='Manuálna')
      AND (cena<=6000)

order by znacka.nazov
```

	Značka	Model	Rok výroby	Palivo	Najazdené	Prevodovka	Cena
1	Citroen	C2	2010	Diesel	49469	Manuálna	5400,00
2	Opel	Astra	2005	Diesel	153000	Manuálna	5300,00
3	Peugeot	Partner	2006	Diesel	113000	Manuálna	2990,00
4	Renault	Scénic	2002	Diesel	177450	Manuálna	2550,00

Tab. 22: Výsledok dopytu Pr. 25

Príklad 26: Zobrazenie aút značky Volkswagen alebo Hyundai, ktoré tankujú benzín a ich rok výroby je rovný alebo nad 2000.

```
select      znacka.nazov as 'Značka', model.nazov as 'Model',
            YEAR(rok_vyroby)  as  'Rok  výroby',  palivo.nazov  as  'Palivo',
            tachometer  as  'Najazdené',  prevodovka.nazov  as  'Prevodovka',
            cena as 'Cena'
from dbo.auto

inner join model on
      model.id_model = auto.id_model
inner join znacka on
      znacka.id_znacka = model.id_znacka
inner join palivo on
      palivo.id_palivo = auto.id_palivo
inner join prevodovka on
      prevodovka.id_prevodovka = auto.id_prevodovka

where (znacka.nazov='Volkswagen' OR znacka.nazov='Hyundai')
      AND (rok_vyroby>='2000')
      AND (palivo.nazov='Benzín')

order by znacka.nazov
```

	Značka	Model	Rok výroby	Palivo	Najazdené	Prevodovka	Cena
1	Hyundai	i30	2007	Benzín	148000	Manuálna	6540,00
2	Hyundai	i30	2007	Benzín	56405	Manuálna	6900,00
3	Volkswagen	Golf	2000	Benzín	225000	Manuálna	3000,00
4	Volkswagen	Polo	2007	Benzín	115000	Manuálna	6000,00

Tab. 23: Výsledok dopytu Pr. 26

Príklad 27: Použitie operátorov *AND* na prepojenie tabuliek typu *INNER JOIN*, čo znamená, že prepojí tabuľky len pre tie údaje, ktoré sa nachádzajú v obidvoch tabuľkách. Pre príklad: Ak v tabuľke Auto bude auto, ktoré nemá určený typ, či je nové, búrané,..., ale má len hodnotu *NULL* (neuvedená hodnota), pri takomto výpise sa nezobrazí.

```
select      znacka.nazov as 'Značka', model.nazov as 'Model',
            YEAR(rok_vyroby)  as  'Rok  výroby', palivo.nazov  as  'Palivo',
            tachometer as 'Najazdené', cena as 'Cena', typs.nazov as
            'Typ'
from  dbo.auto,   dbo.znacka,   dbo.model,   dbo.palivo,   dbo.prevodovka,
      dbo.typs

where  (znacka.id_znacka=model.id_znacka)
      AND  (model.id_model=auto.id_model)
      AND  (palivo.id_palivo=auto.id_palivo)
      AND  (typs.id_typs=auto.id_typs)

order by znacka.nazov
```

	Značka	Model	Rok...	Palivo	Najazdené	Cena	Typ
1	Audi	TT	2007	Benzín	170000	14900,00	Nebúrané
2	BMW	X5	2012	Diesel	41000	24300,00	Nebúrané
3	Citroen	C2	2010	Diesel	49469	5400,00	Nebúrané
4	Citroen	C5	2012	Diesel/Ele...	30	23630,00	Nové
5	Ford	Mondeo	2003	Benzín	212000	3300,00	Nebúrané
6	Hyundai	i30	2007	Benzín	148000	6540,00	Nebúrané
7	Hyundai	i30	2007	Benzín	56405	6900,00	Nebúrané
8	Opel	Astra	2005	Diesel	153000	5300,00	Nebúrané
9	Peugeot	Partner	2006	Diesel	113000	2990,00	Nebúrané
10	Renault	Scénic	2002	Diesel	177450	2550,00	Nebúrané
11	Renault	Kangoo	2011	Diesel	68200	11200,00	Nebúrané
12	Škoda	Fabia	2008	Diesel	32500	6500,00	Nebúrané
13	Volkswagen	Golf	2000	Benzín	225000	3000,00	Búrané
14	Volkswagen	Golf	2005	Diesel	167000	5600,00	Nebúrané
15	Volkswagen	Polo	2007	Benzín	115000	6000,00	Nebúrané
16	Volkswagen	Jetta	1998	Benzín	NULL	500,00	Na náhr...
17	Volkswagen	Touareg	2007	Diesel	160000	10900,00	Nebúrané

Tab. 24: Výsledok dopytu Pr. 27

Príklad 28: Použitie operátora *NOT EXISTS* s podmienkou *IF*. Ak sa v databázi nenachádza zadaná značka auta, zobrazia sa všetky ostatné autá, ktoré sú na výber, ak sa nachádza, tak sa zobrazí len nami zadaná značka auta.

```
IF NOT EXISTS(
                  select * from dbo.auto, dbo.model, dbo.znacka
                  where (znacka.id_znacka = model.id_znacka)
                        AND (model.id_model = auto.id_model)
                        AND (znacka.nazov = 'Škoda')
             )
BEGIN
      select      znacka.nazov as 'Značka', model.nazov as 'Model',
                  YEAR(rok_vyroby) as 'Rok výroby',
                  palivo.nazov as 'Palivo', tachometer as 'Najazdené',
                  cena as 'Cena'
      from dbo.auto

      inner join model on
            model.id_model = auto.id_model
      inner join znacka on
            znacka.id_znacka = model.id_znacka
      inner join palivo on
            palivo.id_palivo = auto.id_palivo;
END
ELSE
BEGIN
      select      znacka.nazov as 'Značka', model.nazov as 'Model',
                  YEAR(rok_vyroby) as 'Rok výroby',
                  palivo.nazov as 'Palivo', tachometer as 'Najazdené',
                  cena as 'Cena'
      from dbo.auto

      inner join model on
            model.id_model = auto.id_model
      inner join znacka on
            znacka.id_znacka = model.id_znacka
      inner join palivo on
            palivo.id_palivo = auto.id_palivo

      where (znacka.nazov = 'Škoda');
END
```

	Značka	Model	Rok výroby	Palivo	Najazdené	Cena
1	Škoda	Fabia	2008	Diesel	32500	6500,00

Tab. 25: Výsledok dopytu Pr. 28

Príklad 29: Dopyt s použitím operátora *EXISTS* pre zobrazenie áut, ktoré sú rezervované zákazníkmi.

```
select      znacka.nazov as 'Značka', model.nazov as 'Model',
            YEAR(rok_vyroby) as 'Rok výroby',
            palivo.nazov as 'Palivo', tachometer as 'Najazdené',
            prevodovka.nazov as 'Prevodovka',
            cena as 'Cena'
from dbo.auto

inner join model on
      model.id_model = auto.id_model
inner join znacka on
      znacka.id_znacka = model.id_znacka
inner join palivo on
      palivo.id_palivo = auto.id_palivo
inner join prevodovka on
      prevodovka.id_prevodovka = auto.id_prevodovka

where EXISTS(select * from rezervacia where id_auto=auto.id_auto)
```

	Značka	Model	Rok výroby	Palivo	Najazdené	Prevodovka	Cena
1	Hyundai	i30	2007	Benzín	56405	Manuálna	6900,00

Tab. 26: Výsledok dopytu Pr. 29

Ak by sme chceli zobraziť autá, ktoré nie sú rezervované zákazníkmi, stačí ak budeme negovať operátor *EXISTS*, tzn., zapísať pred *EXISTS* operátor negácie *NOT*.

Príklad 30: Porovnanie použitia operátora *NOT* a *NOT EXISTS* pre zobrazenie aút, ktoré nie sú búrané.

```
select      TOP 7 znacka.nazov as 'Značka',model.nazov as 'Model',
            tachometer as 'Najazdené',cena as 'Cena',typs.nazov as 'Typ'
from dbo.auto

inner join model on
      model.id_model = auto.id_model
inner join znacka on
      znacka.id_znacka = model.id_znacka
left join typs on
      typs.id_typs = auto.id_typs
```

Použitie operátora *NOT*:

```
where NOT(typs.nazov='Búrané')
```

Použitie operátora *NOT EXISTS*:

```
where NOT EXISTS(
                        select * from auto
                        where typs.id_typs=auto.id_typs
                              AND typs.nazov='Búrané'
                )
order by cena desc, tachometer
```

	Značka	Model	Najazdené	Cena	Typ
1	BMW	X5	41000	24300,00	Nebúrané
2	Citroen	C5	30	23630,00	Nové
3	Audi	TT	170000	14900,00	Nebúrané
4	Renault	Kangoo	68200	11200,00	Nebúrané
5	Volkswagen	Touareg	160000	10900,00	Nebúrané
6	Hyundai	i30	56405	6900,00	Nebúrané
7	Hyundai	i30	148000	6540,00	Nebúrané

Tab. 27: Výsledok dopytu s operátorom *NOT*

	Značka	Model	Najazdené	Cena	Typ
1	BMW	X5	41000	24300,00	Nebúrané
2	Citroen	C5	30	23630,00	Nové
3	Audi	TT	170000	14900,00	Nebúrané
4	Renault	Kangoo	68200	11200,00	Nebúrané
5	Volkswagen	Touareg	160000	10900,00	Nebúrané
6	Audi	A1	152354	9200,00	NULL
7	Hyundai	i30	56405	6900,00	Nebúrané

Tab. 28: Výsledok dopytu s operátorom *NOT EXISTS*

Ako je vidieť, pri použití *NOT EXISTS* sa zobrazí aj auto, ktoré má pri údaje o typu nevyplnené pole. Pri použití *NOT* sa nezobrazí preto, lebo SQL využíva 3-hodnotovú logiku a hodnota *NULL* značí neznámy stav. Ako je známe z 3-hodnotovej logiky, negácia neznámej hodnoty je zasa neznáma. Preto sa nevie určiť v akom stave je typ auta a výsledok zostane nezobrazený.

Pre zobrazenie je použité obmedzenie príkazu *SELECT* funkciou *TOP 7*, preto je pri obidvoch výpisoch zobrazených vždy len sedem áut.

Príklad 31: Ďalšie znázornenie 3-hodnotovej logiky.

Z predchádzajúceho príkladu vieme, že pri neuvedenej hodnote je jej negácia neznáma. Nezobrazí sa nám vo výsledku teda ten riadok, kde nie je uvedený predajca.

```
select predajca,cena from dbo.auto
where NOT predajca='Martin Belanec'
```

	predajca	cena
1	Tomáš Samek	9200,00
2	Lukáš Guček.	6500,00
3	Radek Matula	3000,00
4	Rudolf Škultéty	5600,00
5	Ina Skalica	11200,00
6	Autosalón Krška	24300,00
7	Autokomplex Senica	6000,00
8	Vincent Lukáč	5300,00
9	jokas	6540,00
10	AAA Auto	6900,00
11	AAA Auto	5400,00
12	Pavol Hubek	14900,00
13	Jaroslav Kurejk	10900,00
14	AutoMB	2550,00
15	Ludovít Lego	2990,00
16	Marta Žúrková	500,00

Tab. 29: Výsledok 1. dopytu Pr. 31

Z 3-hodnotvej logiky vieme, že výsledok konjunkcie pravdivej hodnoty a hodnoty neviem je neviem. Z tohto dôvodu sa ani pri nasledujúcom zápise nezobrazí vo výsledku riadok, kde predajca nie je uvedený (*NULL*).

```
select predajca,cena from dbo.auto
where NOT predajca='Martin Belanec' AND cena>2999
```

	predajca	cena
1	Tomáš Samek	9200,00
2	Lukáš Guček.	6500,00
3	Radek Matula	3000,00
4	Rudolf Škultéty	5600,00
5	Ina Skalica	11200,00
6	Autosalón Krška	24300,00
7	Autokomplex Senica	6000,00
8	Vincent Lukáč	5300,00
9	jokas	6540,00
10	AAA Auto	6900,00
11	AAA Auto	5400,00
12	Pavol Hubek	14900,00
13	Jaroslav Kurejk	10900,00

Tab. 30: Výsledok 2. dopytu Pr. 31

Pri použití operátora disjunkcie (*OR*) z 3-hodnotovej logiky je výsledok výrazu pravdivý pre položky, kde je cena áut vyššia ako 2999, teda vo výsledku sa zobrazí aj riadok s neuvedeným predajcom (*NULL*).

```
select predajca,cena from dbo.auto
where NOT predajca='Martin Belanec' OR cena>2999
```

	predajca	cena
1	Tomáš Samek	9200,00
2	Lukáš Guček.	6500,00
3	NULL	3300,00
4	Radek Matula	3000,00
5	Rudolf Škultéty	5600,00
6	Ina Skalica	11200,00
7	Autosalón Krška	24300,00
8	Autokomplex Senica	6000,00
9	Vincent Lukáč	5300,00
10	jokas	6540,00
11	AAA Auto	6900,00
12	AAA Auto	5400,00
13	Pavol Hubek	14900,00
14	Jaroslav Kurejk	10900,00
15	AutoMB	2550,00
16	Ludovít Lego	2990,00
17	NULL	23630,00
18	Marta Žúrková	500,00

Tab. 31: Výsledok 3. dopytu Pr. 31

Ak chceme docieliť, aby sme zobrazili úplne všetky záznamy, teda aj tie, kde nie je uvedený predajca, pri tom to nesmie byť Martin Belanec a cena musí byť vyššia ako 2999 použijeme operátor *NOT EXISTS*.

```
select predajca,cena from dbo.auto a
where NOT EXISTS(
                              select * from dbo.auto b
                              where a.predajca=b.predajca
                              AND b.predajca='Martin Belanec'
                 )
      AND cena>2999
```

	predajca	cena
1	Tomáš Samek	9200,00
2	Lukáš Guček.	6500,00
3	NULL	3300,00
4	Radek Matula	3000,00
5	Rudolf Škultéty	5600,00
6	Ina Skalica	11200,00
7	Autosalón Krška	24300,00
8	Autokomplex Senica	6000,00
9	Vincent Lukáč	5300,00
10	jokas	6540,00
11	AAA Auto	6900,00
12	AAA Auto	5400,00
13	Pavol Hubek	14900,00
14	Jaroslav Kurejk	10900,00
15	NULL	23630,00

Tab. 32: Výsledok 4. dopytu Pr. 31

ZÁVER

Hlavným cieľom tejto bakalárskej práce bolo popísať základne pojmy matematickej logiky a jej využitie v praxi. Matematická logika má uplatnenie v mnohých vedných oboroch a disciplínach, medzi ktoré patria napr. informatika, elektrotechnika, kybernetika, optimalizácia a mnohé iné.

Teoretická časť práce obsahuje pojmy dvojhodnotovej výrokovej logiky, jej algebry a taktiež základné vlastnosti vybraných viachodnotových logík a ich aplikácií.

Praktická časť sa venuje využitiu logických operátorov a funkcií v riešení matematických príkladov, zostavovaní logických obvodov a dopytovania v databázových systémoch.

Pri riešení matematických príkladov sú popísané logické operátory, príkazy softwaru Wolfram Mathematica a ukážky riešených príkladov, kde je znázornené, že logické operátory využívame pri vykresľovaní definičných oborov funkcií. Na základe nich môžeme zapísať logické podmienky pre platnosť danej funkcie alebo pre definovanie oblastí, v ktorých hľadáme viazané extrémy funkcií.

Ďalšou ukážkou matematickej logiky je rozobratie Boolovej algebry pre zostavovanie logických funkcií pre rôzne logické obvody. Názornými príkladmi sme si uviedli, ako zostrojujeme logickú funkciu z pravdivostných tabuliek, ako ju upravíme do minimálneho nami požadovaného tvaru a vytvoríme schému pre zapojenie logického obvodu, ktorý následne túto funkciu realizuje. Taktiež sme si ukázali, že logické hradlá pracujú na základe Boolových funkcií, a že pomocou hradiel *NAND* a *NOR* realizujeme všetky ostatné logické hradlá.

V databázových systémoch sa práca zameriava na použitie logických operátorov a implementáciu trojhodnotovej logiky v dopytovacom jazyku SQL. Presvedčili sme sa, že táto trojhodnotová logika je postavená na pravidlách Łukasiewiczovej logiky a v SQL je využitá pre zavedenie tretej logickej hodnoty *NULL*. Logické operátory v SQL aplikujeme s príkazom *WHERE* a slúžia na presnú špecifikáciu podmnožiny údajov, ktoré majú byť vybrané zo zvolených tabuliek.

Študentom by táto práca mohla pomôcť k porozumeniu matematickej logiky a viesť k lepšiemu pochopeniu, ako na jej základe môžeme realizovať reálne využiteľné systémy a aplikácie.

ZÁVER V ANGLIČTINE

The main objective of this bachelor work is to describe the basic concepts of mathematical logic and its application into practise. Mathematical logic has application in many fields of science and disciplines among which we can include for example computer science, electrical engineering, cybernetics, optimization and many others.

The theoretical section contains the terms of bivalent prepositional logic, its algebra as well as the basic properties of the selected multivalent logics and their application.

The practical part is devoted to the use of the logical operators and functions in solving math problems, compilating of the logic circuits and querying the database systems.

When solving math problems, there are described logical operators, software Wolfram Mathematica commands and examples of the solved math problems where it is shown that the logical operators are used when rendering the domain of functions where on the basis of which we can write logical conditions for the validity of the given function or when defining areas in which we are looking for the committed extremes functions.

Another example of the mathematical logic is to dismantle the Boolean algebra for the construction of the logic functions for the different logic circuits. By means of the illustrative examples we have indicated how to construct a logic function from the truth tables, how to adjust it to the minimum, by us required shape and to create the scheme for the insertion of the logical circuit which subsequently carries out this function. We have also shown that the logic gates operate on the basis of the Boolean functions and that by means of the *NAND* and *NOR* gates we implement all other logic gates.

In the database system, the bachelor work focuses on the use of the logical operators and the implementation of three-valued logic in the SQL querying language. We have seen that the three-valued logic is based on the rules of Łukasiewicz logic and it is used in SQL to introduce a third logic value *NULL*. We apply the logical operators in the SQL by means of the command *WHERE* and these logical operators serve for the precise specification of a subset of data to be selected form the given tables.

This bachelor work could help to the students to understand the mathematical logic and it could lead to the better understanding as to its basis we can realize the real usable system and applications.

ZOZNAM POUŽITEJ LITERATÚRY

[1] ŠRÁMEK, Martin. Zbierka úloh z výrokovej logiky. *Univerzita Komenského v Bratislave: Fakulta matematiky, fyziky a informatiky* [online]. [cit. 2013-03-06]. Dostupné z: http://new.dcs.fmph.uniba.sk/files/texty/ZbierkaUlohZVyrokovejLogiky.pdf

[2] BĚLOHLÁVEK, Radim. Matematická logika: Poznámky k přednáškám. *Univerzita Palackého v Olomouci: Katedra Informatiky* [online]. 14.8.2006 [cit. 2013-03-06]. Dostupné z: http://belohlavek.inf.upol.cz/vyuka/ML.pdf

[3] KOLAŘÍK, Miroslav. Matematická logika. *Univerzita Palackého v Olomouci: Katedra Informatiky* [online]. [cit. 2013-03-06]. Dostupné z: http://phoenix.inf.upol.cz/~kolarikm/MLAI/Pred1.pdf

[4] KVASNIČKA, Vladimír a Jiří POSPÍCHAL. *Matematická logika*. Bratislava: STU vydavateľstvo, 2007. ISBN 8022724491.

[5] OSTRAVSKÝ, Jan a Vladimír POLÁŠEK. *Diferenciální a integrální počet funkce jedné proměnné: Vybrané statě*. Zlín: Univerzita Tomáše Bati vě Zlíně, 2011. ISBN 978-80-7454-124-7.

[6] KAPRÁLIK, Peter, Jana GALANOVÁ a Marcel POLAKOVIČ. *Logické systémy*. Bratislava: STU vydavateľstvo, 2009. ISBN 9788022732055.

[7] Booleova algebra. *Encyklopedie fyziky* [online]. [cit. 2013-03-12]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1322-booleova-algebra

[8] Logické funkce. *Encyklopedie fyziky* [online]. [cit. 2013-03-12]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1323-logicke-funkce

[9] Vícehodnotové logiky. *Filosofický ústav AV* [online]. [cit. 2013-03-16]. Dostupné z: http://logika.flu.cas.cz/files/uploaded/UserFiles/File/svoboda/Vicehodnotove_logiky.pdf

[10] HUDEC, Miroslav, Pavol BÜCHLER a Zuzana RUŽOVIČOVÁ. Výskum, aplikovanie a šírenie fuzzy prístupu pre spracovanie a vyhodnocovanie štatistických dát. *Inštitút informatiky a štatistiky* [online]. [cit. 2013-03-18]. Dostupné z:

http://www.infostat.sk/cevavstat/fuzzy/prispevky/MH-T-CEVAVSTAT0401.pdf

[11] Jak je důležité být fuzzy. *100 vědců do středních škol* [online]. 2012 [cit. 2013-03-24]. Dostupné z: http://www.100vedcu.cz/doc/01-Behounek-Fuzzy.pdf

[12] Fuzzy logika. *Wikipédia: Slobodná encyklopédia* [online]. [cit. 2013-05-07]. Dostupné z: http://sk.wikipedia.org/wiki/Fuzzy_logika

[13] Anti-Lock Brake System. *NFS: Neuro-fuzzy solution* [online]. [cit. 2013-05-11]. Dostupné z: http://neuron.tuke.sk/markuska/nfs/zurnal1.html

[14] Automatická prevodovka. *UNNIP* [online]. [cit. 2013-05-11]. Dostupné z: http://unnip.szm.com/zurnal2.htm

[15] Fuzzy Logic. *Hyundai - Electronics* [online]. 2009 [cit. 2013-05-11]. Dostupné z: http://www.hyundai-electronics.cz/fuzzy-logic-mycky.dic

[16] Automatický systém zaostrovania pre fotoaparáty. *UNNIP* [online]. [cit. 2013-05-11]. Dostupné z: http://unnip.szm.com/zurnal1.htm

[17] VRBA, R., R. KUCHTA, O. SAJDL, P. HUB, M. SKOČDOPOLE, L. FUJCIK, J. HÁZE, M. ZEMÁNEK a R. VRBA. *Multimediální učebnice digitálních obvodů*. Brno: FEKT VUT, 2004. Dostupné z: http://www.umel.feec.vutbr.cz/bdom

[18] ŠEDA, Miloš. *Databázové systémy.* Brno: FSI VUT, 2002. Dostupné z: www.uai.fme.vutbr.cz/~mseda/DBS02_BS.pdf

[19] Structured Query Language. *Wikipédia: Slobodná encyklopédia* [online]. [cit. 2013-05-18]. Dostupné z: http://sk.wikipedia.org/wiki/Structured_Query_Language

[20] Hardwarová realizace logických obvodů. *Mikrokontroléry PIC: Web o číslicové technice a mikrokontrolérech PIC* [online]. 2012 [cit. 2013-05-22]. Dostupné z: http://mikrokontrolery-pic.cz/zaciname/cislicova-technika/hardwarova-realizace-logickych-obvodu/

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

DNF	Disjunctive Normal Form
CNF	Conjunctive Normal Form
SQL	Structured Query Language
DL	Diode Logic
DTL	Diode-Transistor Logic
RTL	Resistor-Transistor Logic
TTL	Transistor-Transistor Logic
CMOS	Complementary Metal-Oxide-Semiconductor
ALU	Arithmetic Logic Unit
CPU	Central Processing Unit
tzn.	To znamená
t.j.	To je
napr.	Napríklad
atď.	A tak ďalej
Tab.	Tabuľka
Obr.	Obrázok
Pr.	Príklad
exk.	Exkluzívny
resp.	Respektíve

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

PI	Prehľad príkazov matematickej logiky	*prehlad.nb*
PII	Zdrojové kódy príkladov 19 – 22	*priklady.nb*
PIII	Vymodelované logické obvody	*logicke_obvody.circ*
PIV	Záloha databázy Autodatabase	*autodatabase_backup*
PV	Použité SQL príkazy	*autodatabase_sql.sql*

Text bakalárskej práce v PDF a všetky prílohy sú vložené na priloženom CD.