StŚedoġkolská technika 2011

Setkání a prezentace prací stŚedoġkolských studentŢ na ĻVUT

VyuġitíprogramŢ pro automatické dokazování v algebŚe

Joel JanļaŚk

Mensa gymnázium, o.p.s.

Španielova 1111/19

163 00 Praha 6 - Řepy

Konzultant: PhDr. Eva Patáková

Praha 2011

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vytvořil samostatně pod vedením PhDr. Evy Patákové a že jsem v seznamu použité literatury uvedl veškeré zdroje, které byly při tvorbě práce využity.

V Praze 24. 1. 2011 Josef Janáček

POD’nKOVÁNÍ

Nejprve bych chtl podkovat vedoucí pŚedložené rolníkové práce Evl Patákové za vynikající pŚístup pŚi konzultacích a mnoho prospšných pŚipomínek.

Dále bych rád podkoval svému otci Antonínu Janlaškovi za pomoc pŚi výbru tématu a mnoho uitelských rad v prbhu práce.

Abstrakt

Roļníková práce studuje jednoduché polookruhy za pomocí programŢ pro automatické dokazování Prover9 a Mace4. Hlavní ļást práce obsahuje hledání koneļných jednoduchých polookruhŢ s pŚedem urļenými vlastnostmi. ZmínĎné programy jsou dále vyuģity v obecném úvodu do algebry. Dále práce nachází také pro program Mace4 dalģí vyuģití.

Abstract

This work studies congruence-simple semirings using programs for automated reasoning Prover9 and Mace4. The main part of this work describes the process of searching for finite congruence-simple semirings with given properties. The aforementioned programs are also used in a general introduction to algebra. Furthermore, the work explores some new uses for the application Mace4.

Obsah

1 Úvod

Má rozšíková práce je zamšená především na využití programů pro automatické dokazování pro konstrukci algebraických struktur s předem danými vlastnosti a dokazování základních vět ve výrokové logice prvního řádu.

Původní motivace práce byla využít programy Prover9 a Mace4 pro studium konečných jednoduchých pologrupů jakožto algebraických struktur. Pologrupy jsou nepříliš prozkoumanou částí algebry, a díky tomu je možné nalézt zde něco nového. Rozhodl jsem se zkusit sestrojit nové konečné jednoduché pologrupy za pomoci programů Prover9 a Mace4. Zmíněné programy jsou blíže popsané v kapitole Použité programy. K těmto programům jsem vymyslel i pár netradičních využití při řešení hlavolamů SUDOKU a zebra. Tím jsem si zároveň procvičil práci s těmito programy v jiné oblasti, než je algebra.

Aby bylo možné porozumět problematice pologrupů, musel jsem se sám obeznámit se základními pojmy algebry. Bez této části by bylo následné pro většinu čtenářů nepochopitelné, a proto jsem vytvořil obecný úvod do této problematiky, který je v kapitole Algebra. V této kapitole předkládám základní definice a snažím se je propojit s různými modely, které vytváří program Mace4. Zároveň zde ukazuji schopnosti programu Prover9, když zkouším vytvořit počítačové důkazy jednoduchých tvrzení. Tyto důkazy se poté snažím zjednodušeně přiblížit čtenářům.

Pro kapitolu Použité programy se kvalitní zdroje hledaly těžko. Malou část jsem použil z internetové stránky programu Prover9 a Mace4, kde o samotných programech příliš napsáno není. Z velké části mi pomohla kniha Automated Reasoning and the Discovery of Missing and Elegant Proofs, která je věnována programu Otter. Program Otter je předchůdcem programu Prover9. O používání programu Mace4 a Prover9 píši hlavně z vlastní zkušenosti.

Pro úvod do algebry jsem zvolil především učebnici profesora Ladislava Procházky Algebra. Když jsem však došel k definici pologrupů, nastal zde menší problém. Definice pologrupu, kterou uvádí Procházka, neodpovídá definici používané v článcích, na kterých jsem založil svou práci, a proto ji nepoužívám. Z tohoto důvodu jsem nemohl použít jiné Procházkovy definice, které na tuto definici navazují. Proto další definice doplňuji ze skript Vektorové prostory od J. Bečváře či Algebra a teoretická aritmetika od J. Novotné. Spíše zřídka jsem nahlédl do slovenského překladu starší knihy Algebra od autorů Saunders Mac Lane a Garrett Birkhoff.

Pro práci s pologrupy jsem používal články z časopisů Journal of Algebra a Journal of Algebra and Its Applications, které vycházely v posledních deseti letech. V těchto článcích byl poprvé učiněn pokus charakterizovat pologrupy po určitých skupinách. Jako další zdroj jsem použil

dizertační práci Ch. Monica, která je zaměřená na využití polokruhů v kryptografii. Dále jsem použil článek V. Flašky na vysvětlení některých pojmů a definic.

Především jsem se rozhodl zkusit sestrojit polookruhy, které charakterizoval Ch. Monico.

2 Použité programy

2.1 Prover9

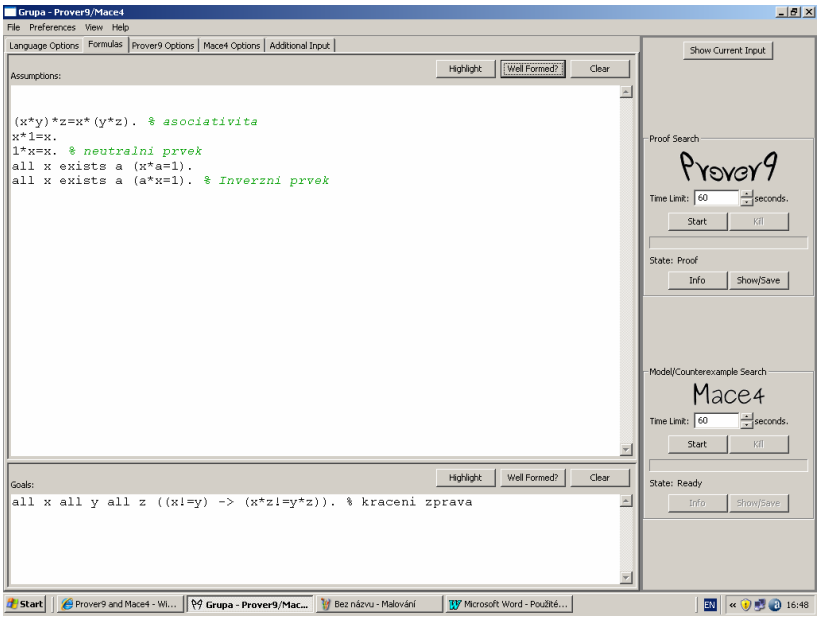
Prover9 je program pro automatické dokazování v matematice a logice. Jeho autorem je William McCune. Tento program vznikl jako nástupce programu OTTER, který vytvořil stejný autor. O programu OTTER vznikla kniha Automated Reasoning and the Discovery of Missing and Elegant Proofs. Tato kniha je pro velkou podobnost programů užitelná i k programu Prover9. Pomocí programu Prover9 jsem vytvářel různé důkazy. Tyto důkazy jsem poté komentoval, neboť poštálová podoba těchto důkazů není obvykle srozumitelná. K programu Prover9 se obvykle přidává program Mace4.

2.2 Mace4

Mace4 je program na hledání modelů a protipříkladů. Pomocí tohoto programu jsem vytvořil všechny tabulky a také jsem pomocí tohoto programu vyhledával určité polookruhy. Další důležitou funkcí Mace4 je tyto modely dále zpracovat pomocí funkce isofiltr. Tuto funkci blíže popíši u izomorfismů.

2.3 Vysvětlení práce s programy Prover9/Mace4

Zde je ukázka, jak vypadá ovládání těchto programů.



Ovládání není příliš složité. Obrázek je ze stránky Formulas, kde se zadávají do Assumptions (horní okno) podmínky. Podmínky se mohou zadávat, tak jak je ukázáno. Program sám rozpozná

symbol * jako binární operaci. Písmena ze zaļátku abecedy pouģívá program jako s existenļním kvantifikátorem a písmena z konce tabulky jako s univerzálním kvantifikátorem. Na prvním Ńádku tedy je: Pro vġechna *x*,*y*,*z* platí (*x***y*)**z*=*x**(*y***z*). V pŃípadļ, ģe se pouģije ļíslo, pak se tímto ļíslem vġdy znaļí pouze urļitý prvek. Z toho plyne, ģe tomuto prvku vlastnļ pŃiŃadíme Ńádek ve výsledné tabulce. V pŃípadļ sloģitļjġích výrokŢ jsem radļji pouģíval pro kvantifikátory all (vġechna) a exists (existuje). Tímto zpŢsobem je na obrázku nadefinovaný inverzní prvek.

V dalġím oknļ (Goals) jsou uvedeny cíle. Tvrzení v tomto oknļ se pokusí Prover9 dokázat. Po napsání zadání staļí jen pustit Prover9 a vytvoŃí se poļítaļový dŢkaz. U komplikovanļjġích dŢkazŢ je vhodné pouģít jeģtļ nastavení v záloģce Prover9 Options. Pokud se dŢkaz nevytvoŃí, je nejjednoduġġí zkusit spustit Mace4. Mace4 se pokusí vytvoŃit protipŃíklad. V pŃípadļ, ģe nalezne alespoŃ jeden model, je jisté, ģe pŢvodní cíl neplatí.

Pro vytváŃení modelŢ s Mace4 je vġak potŃeba jeģtļ upŃesnit zadání, pŃedevġím velikost hledaných modelŢ a jejich poļet v záloģce Mace4 Options.

2.3.1 Prefix, infix a postfix

Zatím jsem operace zadával pouze formou infix. Infix je podoba, kde znak pro operaci je mezi pouģitými prvky. Obvykle se takto pouģívají binární operace jako sļítání, násobení, dļlení a podobnļ. Programy Mace4 a Prover9 jsou vġak samozŃejmļ schopny pouģívat prefix. Operace napsaná jako prefix se oznaļuje pŃed vġemi prvky, které do ní vstupují. Klasickým pŃíkladem je tŃeba nejvļtġí spoleļný dļlitel (D(*x*, *y*)) nebo nejmenġí spoleļný násobek (n(*x*, *y*)). Prefix bývá hojnļ pouģívaný v programování. Já osobnļ zadávám radļji pomocí infixu, i kdyģ prefix mŢģe být u delġích výrazŢ ļitelnļjġí. Postfix je poslední moģný zápis, ļili umístļní znaku pro operaci za prvky. Postfix je velice málo pouģívaná forma zápisu. Jako pŃíklad mļ napadá faktoriál, ale zde nejde o binární operaci. Programy Prover9 a Mace4 neumí pouģívat postfix.

2.3.2 Speciální znaky

Pro poznámky slouģí znak %, který oddļluje zdrojový kód od poznámek. ř ádek se ukonļuje teļkou. K tomuto jeģtļ uvedu, ģe program je schopen pouģívat výrokovou logiku. Ġipka je znak pro implikaci. Pro disjunkci je znak | a pro konjunkci znak &. Pro negaci slouģí znak ï pŃed výrazem. Pomocí tļchto znakŢ vġak není moģné zadat samotnou logiku, neboŃ tyto znaky nevytváŃí operaci, ale pouze danou operaci uplatŃují. Hlavní dŢsledkem toho je, ģe se nezobrazují v koneļné tabulce. A nelze zjistit jejich pravdivostní hodnotu. Pro nerovnost slouģí vykŃiļník pŃed rovnítkem.

2.4 Logika v programech Prover9 a Mace4

Pro vkládání logiky do programů Mace4 a Prover9 je zapotřebí nejprve vložit axiomy logiky. Po vyzkoušení pár možností se mi nakonec osvědčily tyto axiomy. Implikace je *i*, *n* je negace a relace P je kladná pravdivostní hodnota. Operace jsou zadávané prefixem. Zajímavé může být, že poslední axiom je nutný, protože jinak lze prohodit (a program Prover9/Mace4 to tak dělá) pravdivostní hodnotu.

```
P(i(x,i(y,x))).
P(i(i(n(x),n(y)),i(y,x))).
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(P(x)&P(i(x,y)))->P(y).
-P(0).
```

Z těchto axiomů vytvoří Mace4 pouze jediný model. Axiomy logiky by však šly také zadat jiným způsobem. Stačilo by vložit tabulky hodnot.

```
n(0) = 1.
n(1) = 0.

i(0,0) = 1.
i(0,1) = 1.
i(1,0) = 0.
i(1,1) = 1.

- P(0).
  P(1).
```

Takto jsou zadané hodnoty ve formě cooked, kterou Mace4 nabízí ve výstupu. Je to taková forma, která se dá zpětně zadat do vstupu. Tabulku, kterou obvykle ukazuji, není program zpětně schopen přijmout.

Pro lepší zadávání výrazů jsem z těchto axiomů vytvořil klasické logické operace, konjunkci (k), disjunkci (d) a ekvivalenci (e).

```
d(x,y)=(i(n(x),y)).
k(x,y)=(n(d(n(x),n(y)))).
e(x,y)=(k(i(x,y),i(y,x))).
```

Tyto operace budu používat při zadávání výrazů.

Po zadání těchto axiomů stačí napsat složený výrok v závorkách za P a spustit Mace4 s nastavením na dvouprvkové modely (to je pro logiku nutnost, pouze pravda a nepravda). Je vhodné používat písmena ze začátku abecedy (je také vhodné používat velká písmena).

A teď k jednotlivým výsledkům.

1. Pokud nevyjde žádný model, pak to značí, že zadaný výraz nikdy není pravdivý.

2. Pokud vyjde více modelů, pak každý z nich je ŘešeníM. O které jde řešení, je možné zjistit ve výpisu modelů (ukazují se zde hodnoty jednotlivých prvků).

3. Je-li počet řešení roven 2^x, kde x je počet různých prvků ve výrazu, pak jde o tautologii. 2^x je totiž počet všech možných kombinací a tedy maximální počet řešení, a pokud je pro všechny možnosti tento výraz pravdivý, pak je pravdivý vždy. NapříKlad výraz P(e(e(A,B),e(n(A),n(B)))) umožňuje vytvořit čtyři modely, a proto jde o tautologii.

Na podrobnou ukázku jsem vytvořil tento výraz:

P(e(d(A,B),e(n(A),n(C)))).

Vytvoří se čtyři modely. Po použití isofiltru (možnost odstranit izomorfní modely a vypsat si určité prvky, operace apod.) dostáváme toto:

```
interpretation( 2, [number = 1,seconds = 0], [
    function(A, [0]),
    function(B, [0]),
    function(C, [1])]).
interpretation( 2, [number = 2,seconds = 0], [
    function(A, [0]),
    function(B, [1]),
    function(C, [0])]).
interpretation( 2, [number = 3,seconds = 0], [
    function(A, [1]),
    function(B, [0]),
    function(C, [1])]).
interpretation( 2, [number = 4,seconds = 0], [
    function(A, [1]),
    function(B, [1]),
    function(C, [1])]).
```

Jediné, čím se tyto modely odlišují jsou pravdivostní hodnoty prvků A, B a C. Uvedený výrok je pravdivý,

1. když A a B jsou nepravdivé výroky a zároveň výrok C je pravdivý.

2. když výrok B je pravdivý a výroky A a C jsou nepravdivé.
3. když výrok B je nepravdivý a výroky A a C jsou pravdivé.
4. když výroky A, B a C jsou pravdivé.

Možných řešení by bylo 2^3, tj. osm, takže nejde o tautologii.

Pokud chceme zjistit, kdy je výrok nepravdivý, stačí přepsat před P negaci (mínus).

2.4.1 Jednoduchý axiom grupy

Další ze zajímavých využití programu Prover9 a Mace4 může být hledání co nejjednoduššího axiomu. Na ukázku ukážu axiom grupy. Zde je jeden, který uvedl nejnil Kunen: `f(g(f(y,g(y))),f(f(g(y),z),g(f(g(f(y,x)),z))))=x` (Kunen 1995 cit z Wos 2003: str. 281).

Zde je f binární operace a g je unární operace, která k prvku přiřadí jeho inverzní prvek. Že opravdu jde o grupu je schopen Prover9 dokázat.

3 Algebra

3.1 Relace a zobrazení

Tato kapitola je pŚedevęím potŚeba pro vysvĚtlení pojmŢ v kapitole polookruhy. Jedná se hlavnĚ o pojem izomorfní a endomorfní a jeden z nejdŢleęitĚjęích pojmŢ matematiky ï relace.

3.1.1 Relace (binární)

3.1.1.1 Definice

Binární relace R mezi mnoęinami A, B je libovolná podmnoęina R kartézského souęinu mnoęin A, B. Pro dva prvky $a \in A$, $b \in B$ takové, ęe $(a,b) \in R$, píęeme téę aRb a ęteme Ăprvek a je v relaci s (prvkem) bń (Novotná 2004 str. 27).

Relace je tedy libovolné spojení prvkŢ dvou mnoęin do uspoŚádaných dvojic. Toto spojení obvykle plní uręitá pravidla, která je moęno si zvolit. NapŚíklad v kapitole o polookruzích bude hojnĚ uęíván pojem kongruenční relace. Relace mohou být velice uęiteęné pro dŢkazy v algebŚe.

3.1.2 Zobrazení

3.1.2.1 Definice

ĂFunkce F se nazývá *zobrazení tŚídy A do tŚídy B*, jestlięe $\mathrm{Def}^1\ F = A$ a $\mathrm{Im}^2\ F \subseteq B$. Tuto skuteęnost oznaęujeme zápisem $F: A \to B$ nĚkdy téę uęíváme $F: a \to b$, který naznaęuje, ęe typickému prvku $a \in A$ je pŚiŚazen prvek $F(a) = b \in B$ń (Procházka 1990 str. 17).

3.1.3 Homomorfismus, izomorfismus a endomorfismus

3.1.3.1 Definice

ĂBuŇte G, H dva grupoidy. Zobrazení $\varphi: g \to H$ se nazývá *homomorfismus*, jestlięe pro kaędé dva prvky $x,y \in G$, platí $\varphi(xy) = \varphi(x)\varphi(y)$ń (Procházka 1990 str. 60).

Hlavním znakem homomorfismu je tedy, ęe nezáleęí na poŚadí pouęití operace a zobrazení. Toto platí pro vęechny operace. Dalęí definice pŚímo navazuje. Zobrazení φ je tedy homomorfismus:

---

[1] Def znaęí definiční obor

[2] Im od anglického slova image je obor hodnot zobrazení

3.1.3.2 Definice

„Je-li φ bijektivním zobrazením množiny *G* na množinu *H*, pak φ se nazývá *izomorfismus*“ (Procházka 1990 str. 60).

A máme zde první potřebný pojem. Izomorfismus je tedy takové zobrazení, kde pro každý prvek z první množiny je přesně určitelný prvek z druhé množiny (bijektivní zobrazení). Tento výraz byl již zmíněn v popisu program Mace4. Pokud jsou dva grupoidy izomorfní a zároveň nejsou totožné, pak pro každý prvek prvního grupoidu existuje prvek druhého grupoidu. To znamená, že mají pouze přeházené názvy (značení) prvků. Protože značení prvků není důležité (jde o prvky množiny), pak takto vzniklé grupoidy můžeme považovat za stejné. Isofiltr vyřadí takto stejné grupoidy. Totéž lze u isofiltru nastavit i pro konstanty.

3.1.3.3 Definice

„Každý homomorfismus grupoidu *G* do sebe se nazývá endomorfismus grupoidu *G*“ (Procházka 1990 str. 60).

3.2 Struktury s jednou binární operací a jejich důležité vlastnosti

3.2.1 Grupoidy

Grupoid je algebraická struktura s jedinou binární operací.

3.2.1.1 Definice

„Neprázdná množina *G* opatřená binární operací se nazývá *grupoid*“ (Procházka 1990 str. 51).

Konečné grupoidy je možné zapsat Cayleyovou tabulkou. Zde je její názorný příklad. V záhlaví si stačí vybrat prvky, na které se operace vztahuje. Prvek z prvního sloupce je zapisován na levou stranu operace a prvek z horního řádku na stranu pravou. Při pohledu na tabulku zjistíme, že výsledek operace nemusí být vždy stejný. Značení operace se nikdy objevuje ve volném místě v levém horním rohu, například „+“. Takže, zde by to za použití značení plus pro tuto operaci znamenalo 0+0=1, 3+3=0, 2+1=0 atd.

	0	1	2	3
0	1	2	3	0
1	0	1	3	2
2	2	0	0	3
3	1	0	2	0

Každá tabulka vytváří určitou operaci. Jediným požadavkem pro takto napsanou operaci je její uzavřenost. Uzavřenost znamená, že v tabulce jsou ve výsledcích pouze prvky, které jsou v záhlaví (prvky původní množiny). Tento požadavek například nesplňuje obyčejné sčítání na přirozených číslech menších než 5, protože prvek vzniklý součtem 4+2 již není prvkem množiny, která je opatřena binární operací.

3.2.2 Neutrální prvek

3.2.2.1 Definice

„Prvek *e* grupoidu *G* se nazývá *levý* (popř. *pravý*) *neutrální prvek* tohoto grupoidu, jestliže platí:

ea = a pro každé $a \in G$; popřípadě

ae = a pro každé $a \in G$.

Prvek *e* se nazývá *neutrální prvek*, je-li současně levým i pravým neutrálním prvkem.“ (Procházka 1990 str. 52)

V případě použití multiplikativního označení operace lze hovořit o jednotkovém prvku ($(1*x=x)$ & $(x*1=x)$). Pokud použijeme aditivní zápis, tak je možné nazývat neutrální prvek prvkem nulovým ($(x+0=x)$ & $(0+x=x)$). Pro názorné vysvětlení ukážu na tabulce.

	x	1	2	3
x	x	1	2	3
1	1	2	2	3
2	2	x	1	0
3	3	3	x	x

Zde je *x* neutrálním prvkem. Při použití binární operace s neutrálním prvkem *x* bude výsledkem vždy druhý použitý prvek.

3.2.3 Inverzní prvek

Dalším důležitým pojmem je inverzní prvek k danému prvku.

3.2.3.1 Definice

„Nechť *G* je grupoid s neutrálním prvkem *e*. Prvek $b \in G$ se nazývá *levý* (resp. *pravý) inverzní* (či *opačný*) *prvek k prvku* $a \in G$, jestliže *ba=e* (resp. *ab=e*). Je-li *b* současně levým i pravým inverzním prvkem k prvku *a*, tj. *ba=e=ab*, pak *b* se nazývá *inverzním prvkem k prvku a*.“ (Procházka 1990 str. 53).

Inverzní prvek se tedy vyskytuje pouze v grupiodech s neutrálním prvkem. V tabulce se inverzní prvek hledá dobře, stačí najít v tabulce neutrální prvek (ten bývá obvykle značen jedničkou či nulou) a poté najít neutrální prvek ve vnitřní části tabulky.

3.2.3.2 Definice

Grupoid, který má ke každému prvku inverzní prvek, se nazývá grupoid s inverzním prvkem.

V předchozí tabulce máme prvky *x* a 3, které jsou samy sobě inverzním prvkem, prvek 3 je navíc levým inverzním prvkem prvku 2. Prvek 2 je levým inverzním prvkem k prvku 1 a pravým inverzním prvkem k prvku 3. Prvek 1 je pravým inverzním prvkem k prvku 2.
Jak je vidět z předloženého příkladu, pokud je prvek *a* levým inverzním prvkem k prvku *b*, tak zároveň je prvek *b* pravým inverzním prvkem k prvku *a*.
Pokud se jedná o multiplikativní zápis, tak se inverzní prvek obvykle značí jako a^{-1}. V případě aditivního zápisu se inverzní prvek dá napsat jako $-a$.

3.2.4 Anihilující prvek

3.2.4.1 Definice

„Prvek *z* grupoidu *G* se nazývá *levý* (popř. *pravý*) *anihilující prvek*, jestliže platí:
$za = z$ pro každé $a \in G$; popřípadě
$az = z$ pro každé $a \in G$.
Prvek, který je současně levým i pravým anihilujícím prvkem, se nazývá *anihilující prvek*“ (Procházka 1990 str. 54).
V případě multiplikativního zápisu se anihilující prvek označuje jako prvek nulový. V Cayeleyově tabulce není těžké tento prvek nalézt, protože se opakuje na jednom celém sloupci a řádce. V následující tabulce je například anihilující prvek 4.

	0	1	2	3	4
0	2	0	3	0	4
1	0	1	2	3	4
2	3	2	0	2	4
3	0	3	2	3	4
4	4	4	4	4	4

Tato tabulka má ještě neutrální prvek (prvek 1), je asociativní a je komutativní.

3.2.5 Krátitelný prvek

3.2.5.1 Definice

ĂNechŠ G je grupoid a buĎ $a \in G$.

ř íkáme, ěe prvek a je *zleva* (resp. *zprava*) *krátitelný prvek grupoidu G*, jestliěe $ab \neq ac$ (resp. $ba \neq ca$), kdykoliv $b \neq c$ jsou prvky grupoidu G. Jestliěe a je souěasnĚ zleva i zprava krátitelný, pak Říkáme, ěe a je *krátitelný prvek grupoidu G*ń (Procházka 1990 str. 54).

3.2.5.2 Definice

Pokud je kaědý prvek grupoidu G krátitelný, pak se grupoid G nazývá grupoid s krácením.

V pŘípadĚ, ěe jde o grupoid s krácením, se nesmí na ěádném Řádku ani sloupci zápisu do tabulky opakovat jeden prvek dvakrát. To tedy pŘímo vyluěuje anihilující prvek u grupoidu s velikostí vĚtší neě jedna, protoěe tím kdyby se vynásobily dva rŮzné prvky, vznikl by stejný (anihilující) prvek.

3.2.6 DĚlící prvek[3]

3.2.6.1 Definice

ĂNechŠ G je grupoid a $a \in G$.

Je-li $b \in G$, pak Říkáme, ěe a dĚlí b *zleva* (resp. *zprava*), existuje-li $c \in G$ tak, ěe $ac=b$ (resp. $ca=b$)ń (Procházka 1990 str. 54).

3.2.6.2 Definice

Jestliěe a dĚlí kaědý prvek grupoidu, pak se jedná o *dĚlící prvek* tohoto grupoidu.

3.2.6.3 Definice

Pokud je kaědý prvek grupoidu dĚlícím prvkem tohoto grupoidu, pak se jedná o *grupoid s dĚlením*.

3.2.6.4 Tvrzení

Pokud je grupoid s dĚlením a s krácením asociativní, pak jde o grupoid s neutrálním prvkem.

---

[3] Definice pŘevzatá z Procházka 1990, dle Pravidel ěeského pravopisu by bylo lepěí dĚlicí prvek.

3.2.6.5 Důkaz

Pro vyhotovení celého důkazu stačí krácení zleva a dělení zleva. Problém je, že tento důkaz je velice nepřehledný, a to i za použití funkce, která z celého důkazu vytáhne jen tu nejdůležitější část.

```
1 (all x all y all a (x != y -> a * x != a * y)) #
label(non_clause) # label(non_clause).  [assumption].
2 (all x all y exists a x * a = y) # label(non_clause) #
label(non_clause).  [assumption].
3 (exists 0 all a 0 * a = a) # label(non_clause) # label(goal) #
label(non_clause) # label(goal).  [goal].
4 x * (y * z) = (x * y) * z.  [assumption].
5 (x * y) * z = x * (y * z).  [copy(4),flip(a)].
6 x = y | z * x != z * y # label(non_clause).  [clausify(1)].
7 x * f1(x,y) = y # label(non_clause).  [clausify(2)].
8 x * f2(x) != f2(x) # label(non_clause) # label(goal) #
answer(non_clause).  [deny(3)].
10 x * (f1(x,y) * z) = y * z.  [para(7(a,1),5(a,1,1)),flip(a)].
13 x * (y * f2(y)) != x * f2(y) # answer(non_clause).
[ur(6,a,8,a)].
14 $F # answer(non_clause).  [resolve(13,a,10,a)].
```

Na prvních řádcích si Prover9 přepisuje vstupní informace. Čtvrtý a pátý řádek je asociativita (pouze se otáčí rovnost). Na sedmém řádku počítačový důkaz vytváří funkci f1, která je vlastně dělením f1(x,y) = y/x. Dále na osmém řádku vytváří funkci f2, která k potencionálnímu neutrálnímu prvku hledá protipříklad. Vytváří tedy důkaz sporem. V desátém řádku přetváří sedmý řádek, který nejprve vynásobí z, a poté pomocí asociativity umístí z do závorky. V předposledním řádku používá osmý řádek, kde zamění y za x, a to celé pomocí krácení vynásobí x. Z desátého a třináctého řádku nám vznikl spor, protože z z desátého řádku je neutrálním prvkem.

Grupoid s dělením má podobné vlastnosti při zápisu do tabulky jako grupoid s krácením. V zápisu se musí v každém řádku i sloupci vyskytovat každý prvek. Když se podíváme na tabulku, vidíme, že zde není rozdíl mezi grupoidem s krácením a dělením. Pokud nesmí být na jednom řádku či sloupci žádný prvek dvakrát, tak musí být každý prvek tabulky zastoupen právě jednou. Jaký je tedy rozdíl mezi grupoidem s krácením a grupoidem s dělením?

	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

U konečných grupoidů tento rozdíl není. Pokud však vezmeme v úvahu například obor přirozených čísel vzhledem k násobení, zjistíme, že tento grupoid je s krácením, ale nemá každý prvek dělící. Jako druhý příklad si vezmeme grupoid celých vzhledem ke sčítání. Tento grupoid je s krácením i dělením. Pokud však přidáme další prvek, který bude mít stejné vlastnosti jako nula (pouze při sečtení tohoto prvku s jedničkou vznikne opět tento prvek), pak je v skoro každém řádku vždy jeden prvek dvakrát a nejde tedy o grupoid s krácením. Dělení však grupoidu zůstává, neboť na každém řádku i sloupci stále bude každý prvek.

3.2.7 Kvazigrupa

3.2.7.1 Definice

„Grupoid, který je současně s krácením i dělením, se nazývá *kvazigrupa*. Kvazigrupa s neutrálním prvkem se nazývá *lupa*“ (Procházka 1990 str. 55).
Zde je příklad kvazigrupy (bez asociativity a neutrálního prvku).

	0	1	2	3
0	1	0	2	3
1	1	0	2	3
2	1	0	2	3
3	0	2	3	1

Zde je příklad lupy (bez asociativity):

	0	1	2	3
0	1	0	2	3
1	0	1	2	3
2	0	2	1	3
3	0	3	1	2

Z tohoto příkladu plyne, že pokud je grupoid s dělením, krácením a dělícím prvkem, nemusí být asociativní.

3.2.8 Idempotentní prvek

3.2.8.1 Definice

„Prvek *a* grupoidu *G* se nazývá idempotentní čili idempotent, jestliže *a=aa*. Grupoid *G* se nazývá *idempotentní* jestliže každý jeho prvek má tuto vlastnost, tj. platí:
a=aa pro každé *a* ∈ *G*“ (Procházka 1990 str. 55).

3.2.9 Asociativita a komutativita

3.2.9.1 Definice

„Nechť *G* je grupoid. Říkáme, že *G* je *asociativní grupoid* neboli *pologrupa,* jestliže platí následující podmínka, zvaná též *asociativní zákon:*
a(*bc*) = (*ab*)*c* pro všechna *a,b,c* ∈ *G*
Dále říkáme, že *G* je *komutativní grupoid*, platí-li tvz. *komutativní zákon:*
ab=ba, pro všechna *a,b* ∈ *G*“ (Procházka 1990 str. 55).
V případě platnosti asociativního zákona nezáleží na uzávorkování (Bečvář 1989: str. 7). Pokud má pologrupa neutrální prvek, pak jde o monoid.
Komutativní zákon se příklad projeví v tabulce, která dostává osovou souměrnost po diagonále.

	0	1	2	3
0	0	0	2	1
1	0	2	3	1
2	2	3	0	0
3	1	1	0	1

3.2.10 Grupa

3.2.10.1 Definice

„Kvazigrupa, která je současně pologrupou, se nazývá grupa. Jinými slovy, grupa je asociativní grupoid s krácením a dělením“ (Procházka 1990: str. 55).
Lze však najít také jinou definici.
„Množina *G* s binární operací „ se nazývá grupa, jestliže platí následující axiomy:

(i) $\forall a,b,c \in G : \quad (a.b).c = a.(b.c)$,

(ii) $\exists 1 \in G \; \forall a \in G : 1.a = a.1 = a$,

(iii) $\forall a \in G \; \exists a^{-1} \in G : a.\, a^{-1} = a^{-1}.a = 1$.

Jestliže platí jeątĎ axiom

(iv) $\forall a,b \in G : a.b = b.a$,

pak hovoŘíme o komutativní nebo Abelově grupě (BeļváŘ 1989 str. 10).

PŘíkladem Abelovy grupy mŢže být sļítání na oboru celých ļísel nebo násobení na oboru kladných racionálních ļísel.

3.2.10.2 Tvrzení

Obě výše uvedené definice grupy jsou shodné.

3.2.10.3 DŢkaz

Pomocí dŢkazu z kapitoly o dělícím prvku zjistíme, ąe grupa podle první definice musí mít neutrální prvek. A pomocí definice grupoidu s dělením není tĚžké zjistit, ąe má i ke kaądému prvku inverzní prvek. Shodnost definic je jeątĎ potŘeba dokázat opaļně. Toto je dŢkaz, který na asociativním grupoidu s neutrálním a invertním prvkem ke kaądému prvku dokazuje dělení.

```
1 (all x exists a x * a = 1) # label(non_clause).  [assumption].
3 (all a all b exists c a * c = b) # label(non_clause) #
label(goal).  [goal].
4 (x * y) * z = x * (y * z).  [assumption].
6 1 * x = x.  [assumption].
7 x * f1(x) = 1.  [clausify(1)].
9 c1 * x != c2.  [deny(3)].
10 x * (f1(x) * y) = y.
[para(7(a,1),4(a,1,1)),rewrite([6(2)]),flip(a)].
11 $F.  [resolve(10,a,9,a)].
```

DŢkaz je celkem jednoduchý, nejprve se vypíąí vąechny vstupní údaje a pak si je Prover9 upravuje, aą na Řádku deset je sloąí do rovnice, kde pomocí inverzního prvku (funkce f1) nachází pro libovolné x a y prvek (f1(x)*y), takąe nachází vlastně definici dělení. Toto je dŢkaz pro dělení zleva, ale pro dělení zprava je dŢkaz podobný.

Poslední co jeątĎ schází dokázat, je krácení grupy z Procházkovy definice. Zde je jednoduchý dŢkaz:

$a * x = a * y$ /* a^{-1} zleva (to je moąné, neboŘ se obļásti rovnají)

$a^{-1} * (a * x) = a^{-1} * (a * y)$ /asociativita

$1 * x = 1 * y$ /A zde pomocí neutrálního prvku získáváme

$x = y$

Jde tedy o dŧkaz, ŝe v pologrupŏ, kde je neutrální a inverzní prvek, je krácení.

Pro názornou ukázku zde je nejprve Abelova grupa a poté nekomutativní grupa. Mŧŝeme si zde jeŝtŏ ovŏŗit, ŝe kaŝdá grupa má neutrální a inverzní prvek ke kaŝdému prvku a ŝe má krácení i dŏlení. Vŝechny tyto vlastnosti jsou na tabulce lehko pozorovatelné (k tomu jeŝtŏ komutativita v Abelovŏ grupŏ).

	0	1	2	3
0	1	0	3	2
1	0	1	2	3
2	3	2	1	0
3	2	3	0	1

	0	1	2	3	4	5
0	1	0	3	2	5	4
1	0	1	2	3	4	5
2	4	2	1	5	0	3
3	5	3	0	4	1	2
4	2	4	5	1	3	0
5	3	5	4	0	2	1

3.2.10.4 Tvrzení

Druhá definice grupy by se dala zkrátit, a to v bodech dva a tŗi. Pro správné axiomy grupy by staŏ ila v kaŝdém ŗádku pouze jedna rovnost. Uvedenou definici je tedy moŝné zkrátit:

$\forall a,b,c \in G: \quad (a.b).c=a.(b.c)$

$\exists 1 \in G \; \forall a \in G: 1.a=a,$

$\forall a \in G \; \exists a^{-1} \in G: a^{-1}.a=1.$

3.2.10.5 Dŧkaz

Zde je dŧkaz programu Prover9, ŝe pŗedcházející axiomy jsou ekvivalentní s celou druhou definicí. Dŧkaz má dvŏ ŗásti.

```
1 (all x exists a a * x = 1) # label(non_clause).  [assumption].
3 x * 1 = x # label(non_clause) # label(goal).  [goal].
4 (x * y) * z = x * (y * z).  [assumption].
5 1 * x = x.  [assumption].
```

```
6 f1(x) * x = 1.  [clausify(1)].
8 c2 * 1 != c2.  [deny(3)].
9 f1(x) * (x * y) = y.
[para(6(a,1),4(a,1,1)),rewrite([5(2)]),flip(a)].
12 f1(f1(x)) * 1 = x.  [para(6(a,1),9(a,1,2))].
13 f1(f1(x)) * y = x * y.  [para(9(a,1),9(a,1,2))].
14 x * 1 = x.  [back_rewrite(12),rewrite([13(4)])].
15 $F.  [resolve(14,a,8,a)].
```

Na řádcích 1 až 5 vypisuje zadání (na řádku tři je cíl). Na řádku 6 vytváří Prover9 funkci, která vytváří levý inverzní prvek. Poté Prover9 vytváří na osmém řádku negaci cíle, jde tedy opět o důkaz sporem. Devátý řádek je řádek čestný vynásobený zprava *y*. Dvanáctý řádek je předposlední devátý řádek, kde se za *x* dosadí f1(*x*) a za *y* se dosadí *x*. Z toho vyjde za použití čestného řádku řádek dvanáctý. Třináctý řádek jen dokazuje, že f1(f1(*x*)) = *x*. To je podrobněji popsáno v druhé části důkazu (popis k řádku třináct). Pokud dokážeme i toto, vzniká řádek čtrnáct, který má spor s řádkem osm.

```
1 (all x exists a a * x = 1) # label(non_clause).  [assumption].
2 (all x exists a x * a = 1) # label(non_clause) # label(goal).
[goal].
4 (x * y) * z = x * (y * z).  [assumption].
5 1 * x = x.  [assumption].
6 f1(x) * x = 1.  [clausify(1)].
7 c1 * x != 1.  [deny(2)].
9 f1(x) * (x * y) = y.
[para(6(a,1),4(a,1,1)),rewrite([5(2)]),flip(a)].
13 f1(f1(x)) * y = x * y.  [para(9(a,1),9(a,1,2))].
19 x * f1(x) = 1.  [para(13(a,1),6(a,1))].
20 $F.  [resolve(19,a,7,a)].
```

Znovu jsou v prvních řádcích vypsány vstupní údaje a znovu se na řádku čest vytváří funkce f1, která značí x^{-1}. Na řádku sedm se vytváří negace cíle, opět tedy jde o důkaz sporem. Řádek devět je pomocí asociativity přeměněný řádek čest za použití neutrálního prvku. Na třináctém řádku přetváří devátý řádek, tak že dosadí za *x* = f1(*x*) a za *y* = *x* + *y*. Z toho vyjde odstranění jednoho neutrálního prvku rovnice třináct. Zde však musím přiznat, že nerozumím přechodu na rovnici na řádku devatenáct, proto ji nahradím jednoduchým „lidským“ důkazem. Stačí do čestky dosadit za

$x = f1(x)$, a protože máme na Šádku tŠinácť dokázáno, že $f1(f1(x)) = x$, vyjde z toho ihned Šádek devatenáct. Je možné, že toto používá i poľítaľový dŢkaz, ale jisté to není.

SamozŠejmŊje možná i druhá možnost, a to opaľná:

$\forall a,b,c \in G: \ (a.b).c=a.(b.c)$

$\exists 1 \in G \ \forall a \in G: a.1=a$

$\forall a \in G \ \exists a^{-1} \in G: a. a^{-1}=1.$

DŢležité tedy je mít inverzní a neutrální prvek na stejné straně Uvedenou definici nelze zkrátit takto:

$\forall a,b,c \in G: \ (a.b).c=a.(b.c)$

$\exists 1 \in G \ \forall a \in G: 1.a=a$

$\forall a \in G \ \exists a^{-1} \in G: a. a^{-1} =1.$

To dokazuje tento protipŠíklad:

	0	1	2	3
0	0	1	2	3
1	0	1	2	3
2	0	1	2	3
3	0	1	2	3

Nejde o grupu, není zde neutrální prvek.

3.3 Struktury s dvěma binárními operacemi

3.3.1 Obecně

V pŠípadě použití dvou binárních operací se obvykle jedna znaľí aditivně a druhá multiplikativně Takto je možné rozlišit některé definice, a to především nulový prvek. Nulový prvek se takto dá definovat jako neutrální prvek vzhledem ke sľítání a anihilující prvek vzhledem k násobení. Další prvek, který dostává spojením dvou operací smysl, je nekoneľno. Nekoneľno je anihilujícím prvkem pro sľítání i násobení. Z tŊchto dŢvodŢ je nutné dávat si pozor, pro kterou operaci je daný údaj napsán.

3.3.2 Distributivita

3.3.2.1 Definice

Pro vęechny prvky *a*, *b*, *c* platí: $a * (b + c) = (a * b) + (a * c)$ a souļasnď $(b + c) * a = (b * a) + (c * a)$.

Toto je *distributivní zákon*. V algebŚe se pouęívá napŚíklad u okruhŢ. Distributivní zákon tedy vytváŚí vztah mezi dvěma operacemi.

3.3.3 Okruhy

3.3.3.1 Definice

ĂMnoęina R se dvěma binárními operacemi Ă+ñ a Ă*ñ se nazývá okruh, jestliže platí následující axiomy:

$\forall a,b,c: (a + b) + c = a + (b + c)$

$\forall a,b: a + b = b + a$

$\exists 0\ \forall a: a + 0 = a$

$\forall a\ \exists (ï a): a + (ï a) = 0$

$\forall a,b,c: (a * b) * c = a * (b * c)$

$\forall a,b,c: a * (b + c) = (a * b) + (a * c)$

$\forall a,b,c: (b + c) * a = (b * a) + (c * a)$ñ (BeļváŚ 1989 str. 13).

Okruh je spojení dvou operací distributivním zákonem, když jedna z operací vytváŚí Abelovu grupu a druhá je svázaná asociativním zákonem. Obvykle je operace vytváŚející Abelovu grupu sļítání a násobení je svázáno asociativním zákonem.

PŚíkladem okruhŢ jsou napŚíklad tyto ļíselné obory: celá ļísla, racionální ļísla, reálná ļísla a komplexní ļísla.

3.3.4 Polookruh

3.3.4.1 Definice

ĂPolookruh je neprázdná mnoęina s dvojicí asociativních operací, kde násobení je oboustrannď distributivní vzhledem ke sļítáníñ (El Bashir 2001 str. 277 volný pŚeklad).

Více k definici i polookruhŢm celkovď je v kapitole polookruhy.

3.3.5 Těleso

3.3.5.1 Definice

„Obecně *tělesem* rozumíme každou uspořádanou trojici (*T*, *, °), kde *T* je aspoň dvouprvková množina, *, ° jsou operace na *T* a platí (*x*, *y*, *z* ∈ *T*):

(1) $x * y = y * x$, (1´) $x \circ y = y \circ x$,

(2) $(x * y) * z = x * (y * z)$, (2´) $(x \circ y) \circ z = x \circ (y \circ z)$,

(3) $(\exists\, 0 \in T)(\forall\, x)\ 0 * x = x$, (3´) $(\exists\, 1 \in T)(\forall\, x)\ 1 \circ x = x$,

(4) $(\forall\, x)\ (\exists$ ï $x \in T)\ x * (\text{ï}\, x) = 0$,

(4´) $(\forall\, x \neq 0)\ (\exists\, x^{-1} \in T)\ x \circ x^{-1} = 0$,

(5) $(x * y) \circ z = (x \circ z) * (y \circ z)$“ (Novotná 2006 str. 27).

4 Netradiční využití programu Mace4

4.1 SUDOKU

SUDOKU patří v dnešní době k nejoblíbenějším hlavolamům. Vzniklo přibližně v roce 1979 a jeho autorem je Howard Garnsem. SUDOKU je hlavolam, který obsahuje devět číselných řad 1 až 9. Hlavolam SUDOKU je zadán do tabulky o rozměrech 9×9, kde se nachází devět čtverců o rozměrech 3×3, které se nepřekrývají (příklady jsou níže). Některá z čísel jsou zadána a ostatní má řešitel při dodržení následujících pravidel doplnit.

Základní pravidla SUDOKU jsou velmi jednoduchá. To je asi důvod, proč jsou SUDOKU tak oblíbené. V žádném řádku, sloupci ani bloku se nesmí opakovat stejná číslice, což znamená, že volná pole se vyplňují tak, aby se číslice (1 až 9) objevila pouze jednou v každé z devíti řad, v každém z devíti sloupců a v každém z devíti čtverečků.

Podmínky, které utváří tabulku kvazigrupy, mě přivedly na zajímavé využití programu Mace4. Při pohledu na kvazigrupu jsem zjistil, že připomíná SUDOKU. Rozhodl jsem se tedy pokusit se řešit hlavolam SUDOKU pomocí programu Mace4. SUDOKU zde budou vždy s nulou místo devítky. Pokud budeme definovat SUDOKU pouze jako kvazigrupu, pak chybí podmínka, že v každém čtverci třikrát tři nesmí být žádná číslice dvakrát. Proto může program Mace4 nalézt více možných řešení. Zde je příklad SUDOKU (první SUDOKU v přílohách).

8	6	3	2			5		4
5	4	1	3		0			8
					8	1	3	6
	3		1	0			8	
2	5	8	7	3	4	6		
6	1				5		7	
3			6			0		
1		4			3		6	2
0	8		4	1	2	3		

Toto SUDOKU se mi podařilo vyřešit pomocí programu Mace4, který mi po vložení zadání nabídl dva možné výsledky. Jejich rozdíly jsem podtrhl. Zajímavé může být, že v některých menších čtvercích jsou stejné.

8	6	3	2	7	1	5	0	4
5	4	1	3	6	0	7	2	8
7	0	2	5	4	8	1	3	6
4	3	7	1	0	6	2	8	5
2	5	8	7	3	4	6	1	0
6	1	0	8	2	5	4	7	3
3	2	5	6	8	7	0	4	1
1	7	4	0	5	3	8	6	2
0	8	6	4	1	2	3	5	7

8	6	3	2	7	1	5	0	4
5	4	1	3	6	0	7	2	8
4	0	7	5	2	8	1	3	6
7	3	2	1	0	6	4	8	5
2	5	8	7	3	4	6	1	0
6	1	0	8	4	5	2	7	3
3	2	5	6	8	7	0	4	1
1	7	4	0	5	3	8	6	2
0	8	6	4	1	2	3	5	7

Druhý výsledek vůbec nesplňuje podmínku o čtvercích.

Vyřešené SUDOKU má vloženo 42 číslic a řadí se k jednodušším SUDOKU. Obtížnost SUDOKU na počtu vložených číslic příliš nezávisí. Na to poukazuje další příklad. SUDOKU, kde bylo čtyřicet číslic, se podařilo vyřešit pouze s jedním nalezeným řešením (druhé SUDOKU v přílohách). Pro SUDOKU s třiceti čísly (třetí SUDOKU v přílohách) na začátku mi však program nabídl 3268 možných řešení. Zde to SUDOKU je:

	4		5	3			2	8
7					0		1	
5			1	4	7			3
		8			3		6	
		7				0		
	0		6			5		
3			4	0				
	1			6				
4	2			8			7	

Poté se mi osmi nerovnicemi (vždy jsem našel ve stejném čtverečku dvě stejné číslice, jednu ze zadání, druhou jsem zakázal) podařilo snížit počet na 29, z nichž jsem našel hledané SUDOKU. Přesto však je toto hledání velmi zdlouhavé, a proto ho není možné doporučit.

Rozhodl jsem se vytvořit podmínku, které neumožní, aby se stejné číslo opakovalo v malých čtvercích. S touto podmínku je možné vyřešit libovolné SUDOKU bez nutnosti dalšího provádění.

```
s(0)=0.
s(1)=0.
s(2)=0.
s(3)=1.
s(4)=1.
s(5)=1.
s(6)=2.
s(7)=2.
s(8)=2.
(s(x)=s(y)&s(u)=s(v)&(x!=y | u!=v))->x*u!=y*v.
```

Zůstává zde však jeden problém, potíž je s přepisem SUDOKU do formy, kterou Mace4 dokáže přečíst. Nejjednodušší způsob je nechat si přepsat již vyřešené SUDOKU do formy cooked a zde přepisovat čísla, která již jsou zapsaná, a mazat čísla, která v SUDOKU chybí. To znamená projít 81 řádků a každý minimálně zkontrolovat.

4.1.1 Algebraické vlastnosti SUDOKU

U hlavolamu SUDOKU jsem však mohl zadat další podmínky, které by mohly zúžit počet řešení. Pro další text je třeba si uvědomit, že SUDOKU zadávám pomocí operace násobení, a proto jsou souřadnice, které budu uvádět, shodné se zápisem operace. Tedy souřadnice [2,5] znamená vlastně pole 2 * 5.

SUDOKU není komutativní. Pokud by bylo, pak by nemohlo splňovat pravidlo s malými čtverci. Například prvky se souřadnicemi [0,1] a [1,0] by musely být stejné a přitom leží ve stejném čtverci. Další podmínku, kterou jsem mohl zadat je, že SUDOKU nemá neutrální prvek. Pokud by totiž byl jeden prvek a neutrální, pak na souřadnici [a - 1,a] a souřadnici [a,a ï 1], kde bude číslo a ï 1, či souřadnicích [a + 1,a] a [a,a + 1], s číslem a + 1 budou ve stejném čtverci (pokud jedna ze souřadnic není možná, pak zbývající dvojice leží ve stejném čtverci). Další podmínkou je, že SUDOKU není asociativní. To již není vidět na tabulce a ani to nejde jednoduše dokázat přes

Prover9, protože tento program dokazuje pro všechny velikosti modelů a ne jenom pro ty s devíti prvky. Existuje však jednoduchý důkaz, který je uveden v kapitole Algebra. Jde o důkaz, že pokud je kvazigrupa asociativní (grupoid s krácením a dělením), pak musí mít neutrální prvek. Ten však SUDOKU mít nemůže.

4.2 Zebry

Zebra je hlavolam, který se obvykle zadává slovně. Jde vlastně o určitý typ hádanky. Zde je příklad (z internetové stránky http://www.cz-milka.net/logicke-hadanky/skvely-fotbal/ z 26. 12. 2011), na kterém se pokusím zebry blíže popsat:

SKVĚLÝ FOTBAL

Při hráči hraje v různých mužstvech na různých pozicích a má různě barevné dresy.

Hráč Carolina Panthers má fialový dres.
Samuel není zadní útočník, křídelní útočník hraje v mužstvu Dallas Cowboys.
Obránce má žlutý dres, Claude hraje za Dallas Cowboys.
Útočník nehraje za Green Bay Packers.
David nehraje za Oakland Raiders, Samuel je v černém.
Střední obránce je z Oakland Raiders, David hraje ve žlutém.
Victor je z Cleveland Browns a nenosí modrý dres.
Bill je útočník a hráči Cleveland Browns nosí červené dresy.
Určete dres, družstvo a postavení každého hráče.

Stejně jako tato zebra mají i ostatní zebry určitý počet postav, ke kterým se přiřazuje stejný počet předmětů (vlastností apod.) s tím, že ke každé postavě patří vždy pouze jeden předmět. Poté se krátkými údaji spojují dané údaje, či se dané spojení vylučuje. Každá zebra by měla mít pouze jedno řešení.

4.2.1 Jak řešit zebry pomocí programu Mace4

Nejprve je dobré zaměnit dané předměty za čísla. Zde jsem vypsal všechny údaje a přiřadit jim v tomto pořadí čísla 0 až 4.
Tým: Carolina Panthers, Dallas Cowboys, Green Bay Packers, Oakland Raiders, Cleveland Browns
Dres: fialový, žlutý, černý, modrý, červený

Pozice: zadní útoļník, kŚídelní útoļník, obránce, útoļník, stŚední obránce
Jméno: Samuel, David, Viktor, Claude, Bill

Poté jsem pŚepsal podmínky:

```
druzstvo(x)=0 -> dres (x)=0.
jmeno(x)=0 -> pozice(x)!=0.
pozice(x)=1->druzstvo(x)=1.
pozice(x)=2->dres(x)=1.
pozice(x)=3->druzstvo(x)!=2.
jmeno(x)=1->druzstvo(x)!=3.
jmeno(x)=0->dres(x)=2.
pozice(x)=4->druzstvo(x)=3.
jmeno(x)=1->dres(x)=1.
jmeno(x)=2->druzstvo(x)=4.
jmeno(x)=2->dres(x)!=3.
jmeno(x)=3->druzstvo(x)=1.
jmeno(x)=4->pozice(x)=3.
druzstvo(x)=4->dres(x)=4.
```

Chybí jeątŋposlední krok:

```
jmeno(x)=x.
```

To upevní jméno jako základ a poté k nim pŚikládá daląí údaje. Tím se vylouļí více Śeąení se zámŋnou ļíslic.

Poté staļí pouze spustit Mace4 nastavenou na modely s pŋti prvky.

```
    function(dres(_), [2,1,4,3,0]),
    function(druzstvo(_), [3,2,4,1,0]),
    function(jmeno(_), [0,1,2,3,4]),
    function(pozice(_), [4,2,0,1,3])]).
```

TeŅ staļí pouze spojit vąechny první prvky a po pŚepisu jmen zjistíme, ąe Samuel má ļerný dres a hraje stŚedního obránce Oakland Raiders. Stejnŋ spojíme i ostatní prvky a máme vyŚeąenou celou zebru. David má ąlutý dres a hraje obránce u Green Bay Packers. Viktor má ļervený dres a hraje na pozici zadního útoļníka u Cleveland Browns. Claude má modrý dres a hraje na pozici kŚídelního útoļníka u Dallas Cowboys. Bill má fialový dres a hraje na pozici útoļníka u Carolina Panthers.
Tímto zpŢsobem jdou Śeąit vąechny zebry, a to i ty sloąitŋjąí.

5 Polookruhy

Neexistuje jednotná definice polookruhů. První definice tvrdí: Polookruh je neprázdná množina s dvojicí asociativních binárních operací, které spojuje distibutivita. Procházka (1990) k této definici přidává jeětě, že sčítání je komutativní s neutrálním prvkem. Druhá definice polookruhu se od definice okruhu liší pouze tím, že polookruh nevyžaduje neutrální prvek u násobení a inverzní prvek u sčítání. Každý okruh je tedy polokruh. První zmínky o polookruzích se vyskytují na přelomu 19. a 20. století. Definice polookruhů, kterou jsem si pro svou práci vybral, je používaná ve článcích uveřejněných v Journal of Algebra a v Journal of Algebra and Its Aplications, ze kterých jsem především čerpal. Tyto definice jsou v angličtině, proto nikdy uvádím anglický název do závorky. Polookruhy nejsou tak používané jako okruhy a jsou spíše okrajovou oblastí algebry. Polookruhy se používají v aplikované matematice.

5.1 Konečné jednoduché polookruhy

Má práce se zaměřuje na konečné jednoduché polookruhy (*finite congruence-simple semirings*).

5.1.1 Kongruenční relace (congruence relation)

$x \approx y \rightarrow (c + x \approx c + y)$

$x \approx y \rightarrow (x + c \approx y + c)$

$x \approx y \rightarrow (c * x \approx c * y)$

$x \approx y \rightarrow (x * c \approx y * c)$

$x \approx x$

$x \approx y \rightarrow y \approx x$

$(x \approx y) \wedge (y \approx z) \rightarrow (x \approx z)$

Všechny jednoduché polookruhy mají pouze relaci všech prvků se všemi prvky, anebo každý prvek sám se sebou. Tyto polookruhy mají každý prvek výjimečný a nedá se tedy více prvků nahradit jedním. Tato vlastnost jde pozorovat i na tabulce. Pokud se v řádcích a sloupcích některých prvků opakují pouze ty stejné prvky nebo je jejich výsledek pro všechny stejný, pak tyto prvky jsou v kongruenční relaci. Tímto se jednoduché polookruhy stávají jakýmsi stavebním kamenem, lze pomocí nich stavět složitější struktury. Typickým příkladem může být Modulo 5 (zbytek po dělení při) na nezáporných celých číslech. To není jednoduchý okruh, protože všechna čísla se stejným zbytkem po dělení se chovají stejně. Pokud však zadáme operaci modulo 5 na nezáporná čísla, která jsou menší než 5, pak se jedná o jednoduchý okruh.

5.2 Dosavadní výzkum v oblasti jednoduchých polookruhů

Tyto polookruhy se zdají ve srovnání s okruhy a grupami velice neprozkoumané. První charakteristika byla uveřejněna až v roce 2001 (El Bashir 2001). V tomto článku se pozornost také zaměřovala na komutativní konečné jednoduché polookruhy. Byly nalezeny tyto dvouprvkové komutativní polookruhy.

Z_1

+	0	1
0	0	0
1	0	0

·	0	1
0	0	0
1	0	0

Z_2

+	0	1
0	0	0
1	0	0

·	0	1
0	0	0
1	0	1

Z_3

+	0	1
0	0	0
1	0	1

·	0	1
0	0	0
1	0	0

Z_4

+	0	1
0	0	0
1	0	1

·	0	1
0	0	0
1	1	1

Z_5

+	0	1
0	0	0
1	0	1

·	0	1
0	0	1
1	1	1

Pro polookruh S jsou následující tvrzení ekvivalentní:

- S je konečný jednoduchý polookruh.
- S je konečný a ideal-simple.
- S je izomorfní k jednomu z následujících polookruhů:
  - dvouprvkové polookruhy Z_1 Z_2 Z_3 Z_4 a Z_5;
  - konečné těleso;
  - okruh, kde výsledek násobení je vždy nula a jeho velikost je prvočíslo;
  - polookruh (poloteleso) V(*G*), které je definované níže, *G* je konečná Abelova grupa (El Bashir 2001 str. 303 volný překlad).

Ideal-simple je shodné s bi-ideal-simple. Jedná se o netriviální polookruh *S*, který má pouze jeden ideál *I* s velikostí vyšší než dva. Tedy $S = I$ (Flaška 2005).

Je-li podmnožina *I* polookruhu *S* ideál, pak $(S + I) \cup SI \cup IS \subseteq I$ (Flaška 2005).

5.2.1.1 Definice

Když je násobení konečná Abelova grupa *G* a $V(G) = G \cup \{a\}$ znečí vztah mezi *G* a V(*G*) tak, že platí $x * a = a * x = a$ pro všechna $x \in V(G)$. Sčítání se definuje na V(*G*) podle pravidel:

$x + x = x$, $x + y = a$ pro všechna $x,y \in V(G)$ s podmínkou $x \neq y$.

5.2.2 Aditivně komutativní konečné jednoduché polookruhy

Na tento výzkum navazuje Ch. Monico (2004), který charakterizuje aditivně komutativní konečné jednoduché polookruhy (a*dditively commutative finite congruence-simple semirings*), tedy polookruhy, které nemusí být multiplikativně komutativní.

5.2.2.1 Tvrzení

Ať $I = \{1,2,\dots,m\}$, $\Lambda = \{1,2,\dots,n\}$, a $P = (p_{i,j})$ je $n \times m$ matice a prvky 1 a 0 jsou v ní umístěné tak, že v žádném řádku ani sloupci nejsou pouze nuly, a že nejsou dva řádky stejné a ani žádné dva sloupce nejsou stejné. $S = (I \times \Lambda) \cup \{\infty\}$ a je definovaná binární na S, tak, že:

$(i,\lambda) \cdot (j,\mu) = (i,\mu)$ jestliže $p\lambda j = 1$, jinak platí $(i,\lambda) \cdot (j,\mu) = \infty$. S tím, že pro ∞ platí: $(i,\lambda) \cdot \infty = \infty \cdot (i,\lambda) = \infty \cdot \infty = \infty$.

Pak S je jednoduchý polookruh s počtem prvků $mn + 1$. Každý konečný jednoduchý polookruh s anihilujícím prvkem u násobení je izomorfní k jednomu takto sestrojenému (Monico 2004 str. 853 volný překlad).

5.2.2.2 Tvrzení 2

Ať S je konečný aditivně komutativní jednoduchý polookruh, pak platí jedno z následujících:

(1) $|S| = 2$;

(2) $S \cong \mathrm{Mat}_n(F_q)$ pro nějaká konečná těleso F_q a nějaká $n \geq 1$;

(3) S je okruh, kde násobení je vždy nula a jeho velikost je konečná a rovná prvočíslu *(zero multiplication ring of finite prime order*);

(4) S je aditivně idempotentní;

(5) $(S,.)$ je pologrupa jako v tvrzení 2 s anihilujícím prvkem $\infty \in S$ a $S + S = \{\infty\}$ (Monico 2004 str 853 volný překlad).

K tomu ukazuje, že jsou vzhledem ke sčítání komutativní a idempotentní konečné jednoduché polookruhy, které nejsou komutativní vzhledem k násobení (Monico 2004 str. 854):

+	a	1	b
a	a	1	b
1	1	1	b
b	b	b	b

·	a	1	b
a	a	a	b
1	a	1	b
b	a	b	b

Nepodařilo se mu však sestrojit další takovéto polookruhy. Proto jsem se rozhodl na tuto práci navázat za pomocí programu Mace4. Především jsem se zaměřil na polookruhy, které jsou větší

násobení nekomutativní, mají idempotentní prvek ke sčítání, mají neutrální prvek u násobení a sčítání a anihilující prvek u sčítání.

5.2.3 Aditivně komutativní konečné jednoduché polookruhy s nulou

V roce 2008 J. Zumbrägel dokázal úplně popsat aditivně komutativní konečné jednoduché polookruhy s nulou (additively commutative finite congruence-simple semirings with zero). Toho docílil, když dokázal souvislosti mezi jednoduchými polookruhy a hustými podpolookruhy okruhů endomorfismů konečných idempotentních komutativních monoidů (dense subsemirings of endomorphism rings of finite idempotent commutative monoid), kde hustý podpolookruh je definován následovně:

5.2.3.1 Definice

„Nechť M je idempotentní komutativní monoid. Podpolookruh S ⊆ End(M) je nazýván hustý (dense), jestliže pro všechna $a, b \in M$ endomorfismus $e_{a,b} \in End(M)$. Kde $e_{a,b}$ je definováno následovně: Pokud $x + a = a$ ($x \in M$), pak $e_{a,b}(x) = 0$. Pokud $x + a \neq a$ pak $e_{a,b}(x) = b$“ (Zumbragel 2008 volný překlad).

5.2.3.2 Tvrzení

„Nechť R je konečný polookruh s nulou, který není okruh, pak následující je ekvivalentní:

(1) R je jednoduchý.

(2) |R| ≥ 2 nebo R je izomorfní k hustému podpolookruhu S ⊆ End(M), kde (M,+) je konečný idempotentní komutativní monoid“ (Zumbragel 2008 volný překlad).

Pomocí nalezení této spojitosti se podařilo najít velikosti těchto polookruhů. Z tohoto článku jasně vyplývá, že pokud bude mít mnou hledaný polookruh nulu, pak jeho nejmenší netriviální velikost bude 6. Další budou s velikostí 16 a poté 20. To byla velice užitečná informace pro mé hledání.

5.3 Hledání dalších konečných jednoduchých polookruhů

Na tyto články jsem se rozhodl navázat vlastním výzkumem. Především proto, že Ch. Monico nalezl jednoduchý polookruh, který není multiplikativně komutativní a nepovedlo se mu nalézt další. K tomu mi pomáhal program Mace4, který byl vytvořen v roce 2005, a je tedy mladší než Monicovy články.

5.3.1 Podmínka

Nejdůležitější částí práce bylo vytvořit správnou podmínku na hledání pologokruhů. Základní definice pologokruhů není samozřejmě problém (asociativita sčítání a násobení a distribuvita), ale poté jsem musel správně vybrat z článků další podmínky. Jistě hledám aditivně komutativní pologokruh. Z článků Ch. Monica je zřejmé, že mnou hledané pologokruhy budou také aditivně idempotentní. Pokud chci navázat na výzkum Ch. Monica, nesmím také zapomenout na nekomutativitu násobení. Jednoduchost pologokruhu jsem se rozhodl zajistit zkoušku každého pologokruhu z nalezených na podmínkách kongruenční relace.

K těmto podmínkám jsem ještě přidal nulový prvek u sčítání, jednotkový prvek u násobení a anihilující prvek u sčítání. Tyto podmínky mají za cíl zredukovat počet nalezených modelů (odstraňovat izomorfní pologokruhy). Isofiltr totiž lze spustit až po nalezení modelů a kvůli tomu počet modelů, které musí program najít, stoupá obrovským tempem a jejich hledání trvá neúnosně dlouho a i počítači má s těmito daty problém.

5.3.2 Nalezené pologokruhy

Chtěl jsem hledat netriviální pologokruhy, a proto jsem začal své hledání na tříprvkových pologokruzích. Našly se čtyři pologokruhy, a po použití isofiltru zůstal pologokruh, který našel Monico, a ještě jeden k němu duální. Má osově souměrné násobení podle diagonály. Poté jsem začal zkoumat pologokruhy s čtyřmi prvky. Těch se bez isofiltru našlo 24 a po použití isofiltru jich zůstalo 10. Ani jeden z nich však nesplňuje podmínky jednoduchého pologokruhu. Dále přišly na řadu pologokruhy s pěti prvky. Těch prošlo 528 a isofiltr jich ponechal 86. Už zde jsem si začal říkat, že kontrola podmínky, kterou jsem musel dělat u každého nalezeného pologokruhu zvlášť, trvá dost dlouho. Žádný z těchto pologokruhů nebyl jednoduchý. Pokračoval jsem s pologokruhy s šesti prvky. Zde měl alespoň jeden jednoduchý pologokruh být (Zumbragel 2008). Bez isofiltru prošlo 21768 modelů a isofiltr jejich počet zredukoval na 1004. Všechny tyto pologokruhy jsem musel jednotlivě přenášet k podmínkám relace a kontrolovat je. Ve výsledku jsem nalezl dva jednoduché pologokruhy. Jeden pologokruh měl nulový prvek, ale jeden ho neměl. Do dalších pologokruhů s více prvky jsem se už nepouštěl. Šance na úspěch by byla velice malá vzhledem k množství vynaloženého času. Jistotu, že existuje hledaný pologokruh, bych měl až při hledání pologokruhu se šestnácti prvky (Zumbragel 2008). Rozhodl jsem se ale zkusit vytvořit podmínku, která by omezila množství vzniklých pologokruhů.

5.3.3 Podmínka Zákaz kongruenční relace mezi dvojicí prvků

Podmínku pro jednoduchost polookruhu se mi nalézt nepodařilo. Kongruenční relaci totiž nelze zcela odstranit, protože dvě kongruenční relace má každá neprázdná algebraická struktura. Ze stejného důvodu mi nešla kongruenční relace ani jinak omezit. Podařilo se mi však vytvořit podmínku, která vylučuje kongruenční relaci mezi dvojicí prvků. To mi velice zužuje počet vytvořených modelů, ale i tyto modely poté musím ještě zkontrolovat se základními podmínkami relace. Zde uvádím podobu této podmínky:

```
all x all y  (exists z
((x=y)|((((x+z)!=(y+z))  &     (((x+z!=y) | (y+z!=x))  &  ((x+z!=x)
| (y+z!=y)))
                                  |  (((x*z)!=(y*z))  &     ((x*z!=y) |
(y*z!=x))  &  ((x*z!=x) | (y*z!=y)))
                                  |  (((z*x)!=(z*y))  &     ((z*x!=y) |
(z*y!=x))  &  ((z*x!=x) | (z*y!=y))))))).
```

S touto podmínkou se mi podařilo ve velmi krátkém čase najít polookruh, který nalezl Monico. Nově vzniklá podmínka u tří prvkového polookruhu zajišťuje, že je jednoduchý. Poté jsem začal hledat mezi polookruhy s čtyřmi prvky. Těch bylo nalezeno bez isofiltru 32 768 a po jeho použití zůstaly čtyři polookruhy. To v porovnání s hledáním bez této podmínky sice nalezne více modelů, ale po použití isofiltru jich zůstane ménš Jak je to možné? Tato podmínka totiž vytváří vlastní funkci, která může mít více podob, a pro každou funkci se musí vytvořit další izomorfní polookruh. Oproti žádný z nalezených nebyl jednoduchý. Vznikaly zde i relace tří prvků, které podmínka již nezachytí. Když jsem se rozhodl prozkoumat polokruhy s více prvky, nastal zde problém, počet izomorfních řešení začal neúnosně narůstat. Tuto zátěž nebyl počítač schopen zvládnout. Z tohoto důvodu se neosvědčila ani tato podmínka, která je sice pro polookruhy s velikostí 3-5 vhodná, ale pro vyšší se ukázala nepoužitelná.

5.3.4 Jednoduchý polokruh s nulou

Zde udávám podobu polookruhu s ğesti prvky a nulou, který nalezl Zumbragel (2008).

+ :

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	1	1	1	1	5
2	2	1	2	3	4	5
3	3	1	3	3	1	5
4	4	1	4	1	4	5
5	5	5	5	5	5	5

* :

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	0	0	2	3
3	0	3	2	3	2	3
4	0	4	0	0	4	5
5	0	5	4	5	4	5

5.3.5 Nalezený jednoduchý polookruh s nekone*l*nem

Tento polookruh má stejný anihilující prvek u sļítání a násobení.

+ :

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	1	2	3	4	5
2	2	2	2	4	4	5
3	3	3	4	3	4	5
4	4	4	4	4	4	5
5	5	5	5	5	5	5

* :

	0	1	2	3	4	5
0	0	0	0	3	3	5
1	0	1	2	3	4	5
2	2	2	2	4	4	5
3	0	3	5	3	5	5
4	2	4	5	4	5	5
5	5	5	5	5	5	5

6 Závěr

V mé ročníkové práci se mi povedlo netradičně ukázat základy algebry pomocí počítačových programů Mace4 a Prover9. Všechny informace jsem se pokusil přiblížit na tabulce. Dle mého názoru mohou tabulky nabídnout zcela jiný pohled. Při svém studiu jsem se s ukázkou vlastností na tabulce moc nesetkával. Možná je to z důvodu někdy matoucího vyznění těchto tabulek, na které jsem narazil u grupoidů s krácením a grupoidů s dělením.

Poté jsem tyto poznatky používal v jednoduchých důkazech, které vytváří Prover9. Vytvořené důkazy jsem se naučil číst a podrobně je popisuji. Vytvořené důkazy nejsou žádné novinky, především se jedná o vlastnosti grupy.

Programy Prover9 a Mace4 jsem se naučil používat na základní úrovni. Zajímavé je, že jsem používal více program Mace4, který měl být pouze pomocný k programu Prover9.

Využití programů Mace4 a Prover9 pro luštění SUDOKU nebo hlavolamů Zebra je zajímavé, ale nevidím pro něj další využití. Na to je jejich zadávání příliš zdlouhavé. Mají sloužit k bližšímu poznání možností, které tyto programy nabízejí.

Získal jsem také základní poznatky o algebře, které jsem poté uplatnil v nejdůležitější části práce, ve které se mi podařilo navázat na dosavadní výzkum v oblasti polookruhů.

Vyhledávání konečných jednoduchých polookruhů bylo úspěšné. Nalezl jsem zcela nový netriviální jednoduchý aditivně komutativní polookruh s nekonečnem. Tento polookruh otvírá zcela nové možnosti v oblasti polookruhů, kde se zatím podařilo podrobně popsat pouze jednoduché aditivně komutativní polookruhy s nulou. Mnou nalezený polookruh s nekonečnem může pomoci nalézt cestu k úplnému popsání jednoduchých polookruhů.

7 Pouģitá literatura a pouģité internetové zdroje

BEČVÁŘ, J.. *Vektorové prostory I.*. Praha : SPN, 1989. 171 s.

EL BASHIR, R, et al. Simple commutative semirings. *Journal of Algebra*. 2001, vol. 236, s. 277-306.

FLAŠKA,V.; KEPKA, T.; ŠAROCH, J.. Bi-ideal-simple semirings. *Commentationes Mathematicae Universitatis Carolinae*. 2005, vol. 46, No. 3, s. 391-397.

KUNEN, K., The Shortest Single Axioms for Groups of Exponent 4, *Computers and Mathematics and Applications*, 29 (1995) 1-12.

MAC LANE, S.; BIRKHOFF, G.. *Algebra*. 2. vydanie. Bratislava : ALFA Vydavateľstvo technickej a ekonomickej literatúry, 1974. 664 s.

MONICO, C. *Semirings and Semigroup Actions in Public-Key Cryptography*. Indiana, 2002. 66 s. Dizertační práce. University of Notre Dame.

MONICO, C. On finite congruence-simple semirings. *Journal of Algebra*. 2004, vol. 271, s. 846-854.

NOVOTNÁ, J.; TRCH, M.. *Algebra a teoretická aritmetika : Sbírka příkladů 3. část - Základy algebry*. Druhé, opravené. Praha : Univerzita Karlova v Praze, Pedagogická fakulta, 2004. 136 s. ISBN 80-7290-190-7.

NOVOTNÁ, J.; TRCH, M.. *Algebra a teoretická aritmetika : Sbírka příkladů 1. část - Lineární algebra*. 3. Praha : Univerzita Karlova v Praze, Pedagogická fakulta, 2006. 166 s. ISBN 80-7290-252-0.

PROCHÁZKA, L., et al. *Algebra*. Vydání 1. Praha : Academia, 1990. 560 s. ISBN 80-200-301-0.

VANDIVER, H. S. Note on an associative distributive algebra in which the commutative law of addition does not hold. *Bull. Amer. Math. Soc.*. 1936, Number 12, s. 914-920.

WOS, L.; PIEPER, G. W. *Automated Reasoning and the Discovery of Missing and Elegant Proofs*. New Jersey : Rinton Press, 2003. 372 s. ISBN 1-58949-023-1.

ZUMBRAGEL, J. Classification of finite congruence-simple semirings with zero. *Journal of algebra and its applications*. 2008, vol 7, s. 363-377.

W. McCune, "Prover9 and Mace4", http://www.cs.unm.edu/~mccune/prover9, 2005-2010.

http://www.cz-milka.net/logicke-hadanky/skvely-fotbal/ (z 26. 12. 2011).

8 Přílohy

8.1 Přílohy ke kapitole Použité programy

8.1.1 Tabulka logiky

```
n(0) = 1.
n(1) = 0.

i(0,0) = 1.
i(0,1) = 1.
i(1,0) = 0.
i(1,1) = 1.

- P(0).
  P(1).
```

8.1.2 Logika

```
P(i(x,i(y,x))).
P(i(i(n(x),n(y)),i(y,x))).
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(P(x)&P(i(x,y)))->P(y).
-P(0).

d(x,y)=(i(n(x),y)).
k(x,y)=(n(d(n(x),n(y)))).
e(x,y)=(k(i(x,y),i(y,x))).
```

8.1.3 Tautologie

8.1.3.1 Vstup

```
P(i(x,i(y,x))).
P(i(i(n(x),n(y)),i(y,x))).
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(P(x)&P(i(x,y)))->P(y).
-P(0).

d(x,y)=(i(n(x),y)).
k(x,y)=(n(d(n(x),n(y)))).
e(x,y)=(k(i(x,y),i(y,x))).

P(e(e(A,B),e(n(A),n(B)))).
```

8.1.3.2 Výstup (pouze výroky A a B)

```
interpretation( 2, [number = 1,seconds = 0], [
    function(A, [0]),
    function(B, [0])]).
```

```
interpretation( 2, [number = 2,seconds = 0], [
    function(A, [0]),
    function(B, [1])]).
interpretation( 2, [number = 3,seconds = 0], [
    function(A, [1]),
    function(B, [0])]).
interpretation( 2, [number = 4,seconds = 0], [
    function(A, [1]),
    function(B, [1])]).
% isofilter output A B: input=4, kept=4, checks=1, perms=1, 0.03
seconds.
```

8.1.4 Výraz P(e(d(A,B),e(n(A),n(C)))).

8.1.4.1 Vstup

```
P(i(x,i(y,x))).
P(i(i(n(x),n(y)),i(y,x))).
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(P(x)&P(i(x,y)))->P(y).
-P(0).

d(x,y)=(i(n(x),y)).
k(x,y)=(n(d(n(x),n(y)))).
e(x,y)=(k(i(x,y),i(y,x))).

P(e(d(A,B),e(n(A),n(C)))).
```

8.1.4.2 Výstup

```
interpretation( 2, [number = 1,seconds = 0], [
    function(A, [0]),
    function(B, [0])]).
interpretation( 2, [number = 2,seconds = 0], [
    function(A, [0]),
    function(B, [1])]).
interpretation( 2, [number = 3,seconds = 0], [
    function(A, [1]),
    function(B, [0])]).
interpretation( 2, [number = 4,seconds = 0], [
    function(A, [1]),
    function(B, [1])]).
% isofilter output A B: input=4, kept=4, checks=1, perms=1, 0.01
seconds.
```

8.2 Příklady ke kapitole Algebra

8.2.1 Důkaz tvrzení neutrálního prvku

8.2.1.1 Vstup

```
Assumptions:
```

```
x*(y*z)=(x*y)*z.
all x all y all a (x != y -> a * x != a * y).
all x all y exists a x * a = y.
Goals:
exists 0 all a 0 * a = a.
```

8.2.1.2 Výstup

```
% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.06) seconds.
% Length of proof is 11.
% Level of proof is 3.
% Maximum clause weight is 11.
% Given clauses 4.

1 (all x all y all a (x != y -> a * x != a * y)) #
label(non_clause).  [assumption].
2 (all x all y exists a x * a = y) # label(non_clause).
[assumption].
3 (exists 0 all a 0 * a = a) # label(non_clause) # label(goal).
[goal].
4 x * (y * z) = (x * y) * z.  [assumption].
5 (x * y) * z = x * (y * z).  [copy(4),flip(a)].
6 x = y | z * x != z * y.  [clausify(1)].
7 x * f1(x,y) = y.  [clausify(2)].
8 x * f2(x) != f2(x).  [deny(3)].
10 x * (f1(x,y) * z) = y * z.  [para(7(a,1),5(a,1,1)),flip(a)].
13 x * (y * f2(y)) != x * f2(y).  [ur(6,a,8,a)].
14 $F.  [resolve(13,a,10,a)].
```

8.2.2 Definice grupy

8.2.2.1 Vstup

```
Assumptions:
(x*y)*z=x*(y*z).
x*1=x.
1*x=x.
all x exists a (x*a=1).
all x exists a (a*x=1).
Goals:
all a all b exists c a * c = b.
```

8.2.2.2 Výstup

```
% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.05) seconds.
% Length of proof is 8.
% Level of proof is 3.
% Maximum clause weight is 11.
% Given clauses 4.

1 (all x exists a x * a = 1) # label(non_clause).  [assumption].
```

```
3 (all a all b exists c a * c = b) # label(non_clause) #
label(goal).  [goal].
4 (x * y) * z = x * (y * z).  [assumption].
6 1 * x = x.  [assumption].
7 x * f1(x) = 1.  [clausify(1)].
9 c1 * x != c2.  [deny(3)].
10 x * (f1(x) * y) = y.
[para(7(a,1),4(a,1,1)),rewrite([6(2)]),flip(a)].
11 $F.  [resolve(10,a,9,a)].
```

8.2.3 Zkrácení definice grupy

8.2.3.1 Vstup

```
Assumptions:
all x exists a (a*x=1).
(x*y)*z=x*(y*z).
1*x=x.
Goals:
x * 1 = x.
all x exists a (x*a=1).
```

8.2.3.2 Výstup

```
% -------- Comments from original proof --------
% Proof 1 at 0.03 (+ 0.05) seconds.
% Length of proof is 11.
% Level of proof is 5.
% Maximum clause weight is 11.
% Given clauses 6.

1 (all x exists a a * x = 1) # label(non_clause).  [assumption].
2 x * 1 = x # label(non_clause) # label(goal).  [goal].
4 f1(x) * x = 1.  [clausify(1)].
5 (x * y) * z = x * (y * z).  [assumption].
6 1 * x = x.  [assumption].
7 c1 * 1 != c1.  [deny(2)].
9 f1(x) * (x * y) = y.
[para(4(a,1),5(a,1,1)),rewrite([6(2)]),flip(a)].
10 f1(f1(x)) * 1 = x.  [para(4(a,1),9(a,1,2))].
13 f1(f1(x)) * y = x * y.  [para(9(a,1),9(a,1,2))].
14 x * 1 = x.  [back_rewrite(10),rewrite([13(4)])].
15 $F.  [resolve(14,a,7,a)].

Druhá část důkazu:
% -------- Comments from original proof --------
% Proof 2 at 0.03 (+ 0.05) seconds.
% Length of proof is 10.
% Level of proof is 5.
% Maximum clause weight is 11.
% Given clauses 9.
```

```
1 (all x exists a a * x = 1) # label(non_clause).  [assumption].
3 (all x exists a x * a = 1) # label(non_clause) # label(goal).
[goal].
4 f1(x) * x = 1.  [clausify(1)].
5 (x * y) * z = x * (y * z).  [assumption].
6 1 * x = x.  [assumption].
8 c2 * x != 1.  [deny(3)].
9 f1(x) * (x * y) = y.
[para(4(a,1),5(a,1,1)),rewrite([6(2)]),flip(a)].
13 f1(f1(x)) * y = x * y.  [para(9(a,1),9(a,1,2))].
19 x * f1(x) = 1.  [para(13(a,1),4(a,1))].
20 $F.  [resolve(19,a,8,a)].
```

8.3 Protipříklad ke zkrácení grupy (Mace4)

8.3.1.1 Vstup

```
Assumptions:
all x exists a (x*a=1).
(x*y)*z=x*(y*z).
1*x=x.
Goals:
x * 1 = x.
all x exists a (a*x=1).
```

8.3.1.2 Výstup

```
       | 0 1 2 3
    ---+--------
     0 | 0 1 2 3
     1 | 0 1 2 3
     2 | 0 1 2 3
     3 | 0 1 2 3
```

8.4 Přílohy ke kapitole Netradiční využití programu Mace4

8.4.1 SUDOKU 1

8.4.1.1 Vstup

```
all a all b exists c (a*c=b).
all a all b exists c (c*a=b). % deleni

*(0,1) = 6.
*(0,2) = 3.
*(0,3) = 2.
*(0,6) = 5.
*(0,8) = 4.
*(1,0) = 5.
*(1,1) = 4.
*(1,2) = 1.
*(1,3) = 3.
```

```
*(1,5) = 0.
*(1,8) = 8.
*(2,5) = 8.
*(2,8) = 6.
*(3,1) = 3.
*(3,3) = 1.
*(3,4) = 0.
*(3,7) = 8.
*(4,0) = 2.
*(4,1) = 5.
*(4,2) = 8.
*(4,3) = 7.
*(4,4) = 3.
*(4,5) = 4.
*(4,6) = 6.
*(5,0) = 6.
*(5,1) = 1.
*(5,5) = 5.
*(5,7) = 7.
*(5,8) = 3.
*(6,0) = 3.
*(6,3) = 6.
*(6,6) = 0.
*(7,0) = 1.
*(7,2) = 4.
*(7,5) = 3.
*(7,7) = 6.
*(7,8) = 2.
*(8,0) = 0.
*(8,1) = 8.
*(8,3) = 4.
*(8,4) = 1.
*(8,5) = 2.
*(8,6) = 3.
```

8.4.1.2 Výstup

```
Pouze tabulka
 * :
      | 0 1 2 3 4 5 6 7 8
    ---+-----------------
     0 | 8 6 3 2 7 1 5 0 4
     1 | 5 4 1 3 6 0 7 2 8
     2 | 4 0 7 5 2 8 1 3 6
     3 | 7 3 2 1 0 6 4 8 5
     4 | 2 5 8 7 3 4 6 1 0
     5 | 6 1 0 8 4 5 2 7 3
     6 | 3 2 5 6 8 7 0 4 1
     7 | 1 7 4 0 5 3 8 6 2
     8 | 0 8 6 4 1 2 3 5 7
```

```
 * :
      | 0 1 2 3 4 5 6 7 8
    ---+------------------
     0 | 8 6 3 2 7 1 5 0 4
     1 | 5 4 1 3 6 0 7 2 8
     2 | 7 0 2 5 4 8 1 3 6
     3 | 4 3 7 1 0 6 2 8 5
     4 | 2 5 8 7 3 4 6 1 0
     5 | 6 1 0 8 2 5 4 7 3
     6 | 3 2 5 6 8 7 0 4 1
     7 | 1 7 4 0 5 3 8 6 2
     8 | 0 8 6 4 1 2 3 5 7
```

8.4.2 SUDOKU 2

8.4.2.1 Vstup

```
all a all b exists c (a*c=b).
all a all b exists c (c*a=b). % deleni

% number = 1
% seconds = 0

% Interpretation of size 9

*(0,0) = 3.
*(0,1) = 1.
*(0,2) = 6.
*(0,3) = 5.
*(0,4) = 0.
*(0,8) = 8.
*(1,0) = 4.
*(1,2) = 7.
*(1,8) = 0.
*(2,1) = 0.
*(2,4) = 3.
*(2,5) = 4.
*(2,6) = 6.
*(2,7) = 7.
*(3,0) = 7.
*(3,1) = 6.
*(3,3) = 1.
*(3,4) = 5.
*(3,5) = 0.
*(4,0) = 8.
*(4,4) = 2.
*(4,8) = 6.
*(5,0) = 1.
*(5,1) = 3.
*(5,3) = 4.
*(5,7) = 5.
*(5,8) = 2.
```

```
*(6,1) = 2.
*(6,2) = 1.
*(6,4) = 7.
*(6,7) = 6.
*(7,0) = 6.
*(7,1) = 7.
*(7,2) = 8.
*(7,6) = 1.
*(7,8) = 3.
*(8,0) = 5.
*(8,4) = 6.
*(8,5) = 1.
*(8,7) = 2.
```

8.4.2.2 Výstup

```
Pouze tabulka
       | 0 1 2 3 4 5 6 7 8
    ---+-----------------
     0 | 3 1 6 5 0 7 2 4 8
     1 | 4 8 7 6 1 2 5 3 0
     2 | 2 0 5 8 3 4 6 7 1
     3 | 7 6 2 1 5 0 3 8 4
     4 | 8 5 4 7 2 3 0 1 6
     5 | 1 3 0 4 8 6 7 5 2
     6 | 0 2 1 3 7 8 4 6 5
     7 | 6 7 8 2 4 5 1 0 3
     8 | 5 4 3 0 6 1 8 2 7
```

8.4.3 SUDOKU 3 s plnou podmínkou

8.4.3.1 Vstup

```
all a all b exists c (a*c=b).
all a all b exists c (c*a=b).
s(0)=0.
s(1)=0.
s(2)=0.
s(3)=1.
s(4)=1.
s(5)=1.
s(6)=2.
s(7)=2.
s(8)=2.
(s(x)=s(y)&s(u)=s(v)&(x!=y | u!=v))->x*u!=y*v.

*(0,1) = 4.
*(0,3) = 5.
*(0,4) = 3.
*(0,7) = 2.
*(0,8) = 8.
*(1,0) = 7.
```

```
*(1,5) = 0.
*(1,7) = 1.
*(2,0) = 5.
*(2,3) = 1.
*(2,4) = 4.
*(2,5) = 7.
*(2,8) = 3.
*(3,2) = 8.
*(3,5) = 3.
*(3,7) = 6.
*(4,2) = 7.
*(4,6) = 0.
*(5,1) = 0.
*(5,3) = 6.
*(5,6) = 5.
*(6,0) = 3.
*(6,3) = 4.
*(6,4) = 0.
*(7,1) = 1.
*(7,4) = 6.
*(8,0) = 4.
*(8,1) = 2.
*(8,4) = 8.
*(8,7) = 7.
```

8.4.3.2 Výstup

```
   | 0 1 2 3 4 5 6 7 8
---+------------------
 0 | 0 4 1 5 3 6 7 2 8
 1 | 7 6 3 8 2 0 4 1 5
 2 | 5 8 2 1 4 7 6 0 3
 3 | 1 5 8 0 7 3 2 6 4
 4 | 6 3 7 2 5 4 0 8 1
 5 | 2 0 4 6 1 8 5 3 7
 6 | 3 7 6 4 0 1 8 5 2
 7 | 8 1 5 7 6 2 3 4 0
 8 | 4 2 0 3 8 5 1 7 6
```

8.4.4 Zebra

8.4.4.1 Vstup

```
x!=y -> druzstvo(x)!=druzstvo(y).
x!=y -> jmeno(x)!=jmeno(y).
x!=y -> dres(x)!=dres(y).
x!=y -> pozice(x)!=pozice(y).
%CP,DC,GBP,OR,CB
%fialovy,zluty,cerny,modry,cerveny
%ZU,KU,O,U,SO
%Samuel,David,Viktor,Claude,Bill
```

```
druzstvo(x)=0 -> dres (x)=0.
jmeno(x)=0 -> pozice(x)!=0.
pozice(x)=1->druzstvo(x)=1.
pozice(x)=2->dres(x)=1.
pozice(x)=3->druzstvo(x)!=2.
jmeno(x)=1->druzstvo(x)!=3.
jmeno(x)=0->dres(x)=2.
pozice(x)=4->druzstvo(x)=3.
jmeno(x)=1->dres(x)=1.
jmeno(x)=2->druzstvo(x)=4.
jmeno(x)=2->dres(x)!=3.
jmeno(x)=3->druzstvo(x)=1.
jmeno(x)=4->pozice(x)=3.
druzstvo(x)=4->dres(x)=4.
jmeno(x)=x.
```

8.4.4.2 Výstup

```
interpretation( 5, [number = 1,seconds = 0], [
    function(dres(_), [2,1,4,3,0]),
    function(druzstvo(_), [3,2,4,1,0]),
    function(jmeno(_), [0,1,2,3,4]),
    function(pozice(_), [4,2,0,1,3])]).
```

8.5 Příklady ke kapitole Polookruhy

8.5.1 Tříprvkové

8.5.1.1 Vstup

```
x*(y*z)=(x*y)*z.
x+(y+z)=(x+y)+z. % asociativita
x+y=y+x. % komutativita scitani
x*(y+z)=(x*y)+(x*z).
(y+z)*x=(y*x)+(z*x). %distributivita
x+x=x. % idempotentni scitani
0+x=x. % nulovy prvek scitani
1*x=x.
x*1=x. % jednotkovy prvek nasobeni
x+2=2. % anihilujici scitani potreba menit
exists d (exists e (d*e!=e*d)). % neni komutativni
```

8.5.1.2 Výstup

```
Po použití isofiltru
% number = 1
% seconds = 0

% Interpretation of size 3

 + :
       | 0 1 2
    ---+------
```

```
     0 | 0 1 2
     1 | 1 1 2
     2 | 2 2 2

 * :
       | 0 1 2
    ---+------
     0 | 0 0 0
     1 | 0 1 2
     2 | 2 2 2
% number = 2
% seconds = 0

% Interpretation of size 3

 + :
       | 0 1 2
    ---+------
     0 | 0 1 2
     1 | 1 1 2
     2 | 2 2 2

 * :
       | 0 1 2
    ---+------
     0 | 0 0 2
     1 | 0 1 2
     2 | 0 2 2
```

8.5.2 Polokruhy s velikostí ţ ty

8.5.2.1 Vstup

Stejný jako u tříprvkových, pouze změna u anihilujícího prvku (o +1). Stejně tak i pro další polokruhy. A poté podmínka kongruenční relace:

```
r(x,y) -> r(c + x, c + y).
r(x,y) ->r(x + c, y + c).
r(x,y) ->r(c * x,c * y).
r(x,y) -> r(x * c, y * c).
r(x,x).
r(x,y) -> r(y,x).
(r(x,y)&r(y,z))->r(x,z).
```

Tato relace smí mít pouze dva modely.

8.5.2.2 Výstup

Po podmínce s relací žádné.

8.5.3 Polokruhy s velikostí pět

8.5.3.1 Vstup

Stejný jako u polookruhů s velikostí čtyři.

8.5.3.2 Výstup

Po použití podmínky s relací žádný.

8.5.4 Polookruhy s velikostí šest

Přímo v kapitole Polookruhy.

8.5.5 Podmínka Zákaz kongruenční relace mezi dvojicí prvků

```
all x all y  (exists z
((x=y)|((((x+z)!=(y+z))  &    (((x+z!=y) | (y+z!=x))  &  ((x+z!=x)
| (y+z!=y)))
                              |  (((x*z)!=(y*z))  &    ((x*z!=y) |
(y*z!=x))  &  ((x*z!=x) | (y*z!=y)))
                              |  (((z*x)!=(z*y))  &    ((z*x!=y) |
(z*y!=x))  &  ((z*x!=x) | (z*y!=y))))))).
```