MAGAZÍN
KATEDRY INFORMATIKY

číslo 4 | prosinec 2015 Univerzita Palackého v Olomouci

Úvodní slovo

Vážení čtenáři,

dovolte, abychom se s vámi opět podělili o informace z katedry informatiky a ze světa informatiky vůbec.

Letos je tomu již 50 let od první publikace o fuzzy logice – článku Lotfiho Zadeha *Fuzzy sets*. Při příležitosti tohoto výročí vám přinášíme článek o fuzzy logice, ve kterém se dočtete o tom, proč fuzzy logika vznikla, naleznete v něm průřez její padesátiletou historií i popis jejích komerčně úspěšných aplikací. Dozvíte se také o výzkumu, který v oblasti fuzzy logiky provádějí členové naší katedry.

V druhém článku se zabýváme datovými typy, které jsou nekonečné a dokonce i jejich jednotlivé prvky mohou být nekonečné. Takovým typem jsou například reálná čísla, která bývají reprezentována pouze s určitou přesností čísly racionálními. Reálná čísla lze ovšem v počítači reprezentovat jako funkce. V článku se dočtete podrobnosti o této reprezentaci i o jejích překvapivých vlastnostech.

Pro zájemce o aktuální dění na katedře jsme připravili článek o navazující magisterské Aplikované informatice, nově akreditovaném oboru, který otevíráme v akademickém roce 2016/17. Nechybí samozřejmě ani hádanka, opět programovací.

Za tým, který toto vydaní magazínu připravil, vám příjemné čtení přeje

Petr Osička

redaktor Magazínu Katedry informatiky Univerzity Palackého v Olomouci

Obsah

Nový studijní obor

Jan Outrata

Před rokem, ve druhém čísle magazínu, jsme čtenářům představili všechny studijní obory, které lze na naší katedře studovat. V létě tohoto roku jsme ovšem tuto nabídku kvůli sílící poptávce rozšířili o další studijní obor. Je jím navazující magisterský obor Aplikovaná informatika.

Aplikovaná informatika, navazující magisterská

Tento nový obor lze studovat v prezenční formě studia počínaje akademickým rokem 2016/17. Podobně jako u stejnojmenných oborů v bakalářské etapě studia jsou u nového oboru oproti stávajícímu navazujícímu magisterskému oboru Informatika v menší míře zastoupeny teoretické předměty a větší důraz je kladen na předměty praktického zaměření. Studium je zaměřeno na získání praktických znalostí a dovedností v oblastech informatiky.

Jaká je tedy nabídka? Rámec povinných předmětů tvoří předměty známé z oboru Informatika: algoritmy a složitost, paralelní, distribuované a multimediální systémy, bezpečnost počítačových sítí, moderní webové a objektově orientované technologie a vývoj software. Nutno ovšem podotknout, že většina z těchto předmětů je u oboru Informatika počínaje ak. rokem 2016/17 vedena jako nepovinné, neboť tvoří základ oboru nového.

Nový je balík povinně volitelných předmětů, jejichž absolvováním si student určuje svou specializaci. Nabídka je zde pestrá. Máme tu několik předmětů převzatých z oboru Informatika, například analýza a zpracování obrazu, správa linuxového a Windows serveru nebo soudobé operační systémy. Především se zde ale objevují předměty zcela nové, namátkou komprese dat, platforma Java, tvorba mobilních aplikací nebo paralelní programování v .NET. Navíc budou některé tyto předměty vyučovány odborníky z firem. Po celou dobu studia (2 roky) student také pracuje na své diplomové práci na téma zadané pracovníky katedry nebo firmou.

Absolventi nového oboru budou vysoce kvalifikovanými odborníky nejen ovládajícími moderní informační technologie a metody vývoje software, ale díky znalosti obecných principů informatiky i schopnými tyto technologie a metody analyzovat a přizpůsobovat aktuálnímu vývoji v IT. Uplatní se například jako návrháři informačních systémů, vedoucí softwaroví vývojáři a analytici, programátoři specialisté, odborníci na bezpečnost počítačových sítí a informačních technologií nebo IT manažeři a konzultanti.

Dodejme, že uchazeči, absolventi bakalářského studia informatického nebo příbuzného oboru, budou do studia nového oboru přijímáni na základě úspěšného složení ústní přijímací zkoušky z oblastí informatiky studovaných v bakalářském oboru Aplikovaná informatika.

Další informace ke studiu nového i všech ostatních studijních oborů na naší katedře naleznete na informatika-olomouc.cz a www.inf.upol.cz.

A co Informatika?

Jak již bylo zmíněno, stávající navazující magisterský obor Informatika doznal změn a ty zahrnují i nové předměty v tomto oboru. Bude více orientován na pokročilejší partie a principy informatiky a získání vysoce odborných dovedností založených na hlubších teoretických znalostech. Tyto změny spolu s novým oborem Aplikovaná informatika významně přispějí k výraznému odlišení a rozdílnému zaměření obou oborů. Každý z nich si jistě najde své zájemce!

Fuzzy logika

Radim Bělohlávek

Fuzzy logika nalezla uplatnění v mnoha oblastech lidského konání. Při příležitosti padesátiletého výročí publikace první práce o fuzzy logice přinášíme článek o vzniku, vývoji a aplikacích fuzzy logiky a o výzkumu ve fuzzy logice prováděném na katedře informatiky. Podrobné informace může čtenář najít v obsáhlé knize Fuzzy Logic and Mathematics: A Historical Perspective *autorů R. Bělohlávka, J. W. Daubena a G. J. Klira, kterou vydá nakladatelství Oxford University Press v New Yorku.*

Zadehův nápad

Před padesáti lety zveřejnil časopis *Information and Control* práci *Fuzzy sets* amerického inženýra a matematika Lotfiho Zadeha (*1921). V práci její autor upozornil na to, že základní pojem klasické matematiky – pojem množiny – je v mnoha situacích nedostačující. Množiny reprezentují to, čemu běžně říkáme soubory nebo seskupení nějakých objektů. Seskupeným objektům se říká prvky dané množiny. Tak například máme množinu sudých čísel, jejímiž prvky jsou sudá čísla, nebo množinu států Spojených států amerických, která má padesát prvků. Charakteristickým rysem množin je, že libovolný prvek do dané množiny buď patří, nebo nepatří. Zadeh upozornil na to, že seskupení, se kterými v běžném životě pracujeme, tento rys postrádají a jsou zásadně jiná.

Tvoří například hodnoty (ve stupních Celsia), odpovídající pojmu „vysoká venkovní teplota“ množinu? Do ní by jistě patřila hodnota 35 a naopak nepatřila hodnota 5. Musela by ale existovat hraniční hodnota t ostře oddělující vysoké teploty. Tedy hodnota t taková, že pro libovolně malou odchylku ε platí, že teplota $t-\varepsilon$ není vysoká, zatímco teplota $t+\varepsilon$ vysoká je. To je ale absurdní (představme si třeba $\varepsilon = 0.1$) – tak přece člověk pojem „vysoká venkovní teplota“ nechápe.

Zadehův přínos je možné spatřovat ve třech rovinách. Za prvé, všiml si, že klasický pojem množina je v situacích podobných výše popsané neadekvátní, a proto nedostatečný. Za druhé, ukázal, že z hlediska aplikací je toto pozorování zcela zásadní, že tedy nejde o nějaký okrajový jev, o anomálii, která je zajímavá jen z akademického hlediska. Výrazy jako „vysoká venkovní teplota“, „nízký krevní tlak“, „nadváha“ a podobně – tvrdil Zadeh – jsou v našem každodenním uvažování a rozhodování všudypřítomné a nevyhnutelné. Za třetí, navrhl nový pojem – pojem fuzzy množiny – který zmíněné nedostatky nemá. Základní myšlenka spočívá v tom, že příslušnost prvku k fuzzy množině je otázkou míry – nemá povahu buď-anebo rozhodování, a není tedy „černobílá“.

Po padesáti letech lze konstatovat, že Zadehův nápad byl neobyčejně úspěšný. Zadehova práce o fuzzy množinách z roku 1965 patří mezi nejcitovanější práce v historii matematiky, logiky a informatiky a vedla k rozvoji *fuzzy logiky* jako nové vědní oblasti, která má nejen propracované teoretické základy a na nich založené metody, ale může se pochlubit i obrovským komerčním úspěchem. Jak uvidíme, fuzzy logika představuje významné nové paradigma a s trochou nadsázky lze říci, že je všude kolem nás.

Co je fuzzy logika

Nevhodnost pojmu klasická množina, kterou jsme nahlédli výše, plyne z toho, že výroku „objekt x patří do množiny A“ je klasická matematika v souladu s principy

Zakladatel fuzzy logiky Lotfi Zadeh v roce 2004.

klasické logiky ochotna přiznat pouze dvě možné pravdivostní hodnoty – *pravda* a *nepravda*, někdy označované 1 a 0. Takový výrok, stejně jako každý jiný výrok, může tedy být buď pravdivý, nebo nepravdivý. Fuzzy logika tento princip, zvaný *princip bivalence*, nepřijímá. Podle fuzzy logiky může být výrok pravdivý i jen do určité míry, tj. pravdivý v určitém stupni, který je mezi 0 a 1, např. 0.8. Stupně pravdivosti představují zobecněné pravdivostní hodnoty, přičemž klasické pravdivostní hodnoty 0 a 1 jsou jejich hraničními případy. Tedy například zatímco výrok „35°C je vysoká okolní teplota“ má v souladu s předchozí úvahou pravdivostní hodnotu 1 a výrok „5°C je vysoká okolní teplota“ hodnotu 0, výroku „25°C je vysoká okolní teplota“ můžeme přisoudit pravdivostní hodnotu 0.8. Tím vyjádříme, že teplota 25°C sice není úplně vysoká, ale téměř ano.

Tak lze hovořit o fuzzy množině A vysokých teplot. Pro každou hodnotu x můžeme uvažovat výrok „x°C je vysoká okolní teplota“ a jeho pravdivostní hodnotu označit $A(x)$. Tedy například $A(35) = 1$, $A(5) = 0$, $A(25) = 0.8$ a podobně. Formálně se právě takto definované přiřazení pravdivostních hodnot $A(x)$ hodnotám nebo jiným objektům x nazývá fuzzy množinou. Fuzzy množiny, na rozdíl od klasických, nejsou černobílé, nemají ostré hranice – mezi možnostmi „být prvkem“ (1) a „nebýt prvkem“ (0) je celá škála mezilehlých možností, totiž „být prvkem v jistém stupni“ (např. ve stupni 0.8). Tyto mezilehlé možnosti u klasických množin nejsou.

Odmítnutí principu bivalence, tj. připuštění možnosti, že existují výroky, které nemusí být (úplně) pravdivé, ani (úplně) nepravdivé, je radikální krok. Tímto krokem totiž opouštíme svět klasické logiky důvěrně známý od doby Aristotela a dalších velikánů starověkého Řecka, kde klasická logika jako samostatná disciplína vznikla. Opouštíme tím také svět, ve kterém se od antiky rozvíjela veškerá matematika i další obory, přírodovědné i humanitní, ale i celá západní filozofie. Nový svět, do kterého tímto krokem vstupujeme, je světem jiné logiky. Této logice se říká *fuzzy logika*. Zda je fuzzy logika životaschopnou, plnohodnotnou alternativou logiky klasické a zda se může stát, podobně jako se stala logika klasická, základem matematiky – tentokráte však matematiky jiné – a metodickým základem dalších oborů, je zásadní, mnohovrstevná a značně složitá otázka.

Z historie fuzzy logiky

První upozornil na skutečnost, že princip bivalence není samozřejmý a že některým výrokům – například těm o budoucnosti – je přirozené přiznat i jinou pravdivostní hodnotu než *pravda* a *nepravda*, ve svých klasických spisech *De interpretatione* a *Categoriae* sám zakladatel klasické logiky, Aristotelés. Až do přelomu 19. a 20. století se však úvahy o logice s více pravdivostními hodnotami objevovaly jen ojediněle. Důležitou výjimkou byl významný středověký myslitel William Ockham, který analyzoval Aristotelovy práce a odvodil z nich mimo jiné jistou spojku implikace se třemi pravdivostními hodnotami. Aristotelovy úvahy o možné třetí pravdivostní hodnotě se staly předmětem vášnivých, deset let trvajících debat na univerzitě v Lovani, které se odehrávaly v letech 1465–1475. Za další mezník lze považovat úvahy o vágnosti výrazů přirozeného jazyka. Za vágní jsou označovány právě ty výrazy, které nemají ostře vymezené hranice, jako např. výše uvedený výraz „vysoká teplota“. Tyto úvahy se objevily u vý-

znamného empiristy Johna Lockea, například v jeho základním filozofickém díle *Esej o lidském chápání*. Úvahy o vágních výrazech se poté staly součástí knih o logice – zmiňme první takovou, významnou učebnici *Logick* Isaaca Wattse z roku 1724 – nicméně součástí, které byla zpravidla věnována jen okrajová pozornost.

Na přelomu 19. a 20. století se vícehodnotovými logikami zabývali tři myslitelé: skotský matematik Hugh MacColl, americký filozof a matematik Charles Sanders Peirce, a ruský filozof Nikolaj Vasiljev. Vznik moderní vícehodnotové logiky je však spjat s obdobím kolem roku 1920, kdy své objevy učinili tři význační logici. Jsou to polský logik Jan Łukasiewicz, švýcarský matematik Paul Bernays a americký matematik polského původu Emil Post. Łukasiewicz byl motivován Aristotelovými úvahami o pravdivosti výroků o budoucnosti a navrhl logické systémy se třemi i více pravdivostními hodnotami. Ty jsou známy jako Łukasiewiczovy logiky a patří dnes mezi nejdůležitější systémy fuzzy logiky. Bernays, jehož přínos na poli vícehodnotové logiky byl donedávna neznámý, i Post byli motivováni především matematickými úvahami a problémy. Je ale třeba zmínit, že rozšířený názor, podle kterého Postova práce byl pouhý formalismus, je nesprávný, neboť Post viděl vícehodnotovou logiku jako koncepčně významnou alternativu logiky klasické.

Zejména Łukasiewiczova práce vedla ke značnému rozvoji logik s více pravdivostními hodnotami, a to jak samotné Łukasiewiczovy logiky, tak logik jiných. Ty se liší používanými pravdivostními hodnotami, ale také tím, jak definují logické spojky. Předpokládejme třeba, že výrok φ (např. „Teplota je vysoká“) je pravdivý ve stupni $a = 0.9$ a výrok ψ (např. „Otáčky větráku jsou nízké“) ve stupni $b = 0.8$. Podle pravidel Łukasiewiczovy logiky je výrok $\varphi\&\psi$ (tj. „Teplota je vysoká a otáčky větráku jsou nízké“) pravdivý ve stupni $\max(0, a+b-1) = 0.7$, zatímco podle pravidel tzv. Gödelovy logiky – další dnes významné fuzzy logiky – je to $\min(a, b) = 0.8$. Některé práce přitom vycházely z praktických motivací, jiné se o motivace a případné aplikace nezajímaly a studovaly vícehodnotové logiky jako formální systémy bez ohledu na jejich využití. K těm prvním patří práce Łukasiewicze, který interpretoval třetí pravdivostní hodnotu, $1/2$, jako hodnotu výroku, který je možný (tj. nikoli nutně pravdivý nebo nepravdivý), dále pak práce sovětského logika Dmitrije A. Bochvara, který rozvinul logiku pro analýzu paradoxů, nebo amerického logika Stephena C. Kleeneho, který byl motivován některými otázkami teorie výpočtů. Těch druhých, tj. prací formalistických, které nehleděly na praktické motivace, byla však naprostá většina a jejich podíl rostl. To vedlo k tomu, že se začaly objevovat kritické hlasy – mají vícehodnotové logiky hlubší smysl nebo jsou to jen matematicko-logické hračky?

Na poli filozofie se mezitím, ovšem většinou bez kontaktu s výzkumem ve vícehodnotové logice, začala intenzívně rozvíjet debata o vágnosti. Jedním ze zásadních impulsů byl článek *Vagueness* prominentního matematika a filozofa Bertranda Russella z roku 1923. Brzy se objevily další, mezi nimi také vlivný článek *Vagueness: an exercise in logical analysis* amerického filozofa Maxe Blacka, ve kterém byl – téměř 30 let před Zadehovým článkem, byť v mírně jiné podobě a pod jiným názvem („consistency profile“) – pojem fuzzy množiny v podstatě zaveden. Prací o vágnosti poměrně rychle přibývalo, nepřesáhly však pole filozofie a až na několik výjimek, mezi které patří zmíněná Blackova práce, se otázkou, jak vágnost matematicky uchopit, natož pak, jak případné matematické modely vágních výrazů použít, nezabývaly.

Zadehův článek *Fuzzy sets* zveřejněný v roce 1965 představuje na poli vícehodnotové logiky zásadní milník. Zadeh sice v článku nevytvořil žádnou konkrétní logiku v moderním slova smyslu – ostatně termín „fuzzy logic“ poprvé použil v poněkud technickém smyslu v roce 1966 Marinos a v dnes běžném smyslu až v roce 1968 Goguen. Existujících prací na poli vícehodnotové logiky, ani na poli vágnosti si Zadeh nebyl vědom (na žádnou z nich se v článku neodvolává). Hnacím motorem pro něho byl zmíněný nedostatek klasického pojmu množina – neschopnost reprezentovat význam vágních výrazů, které jsou typické pro přirozený jazyk a zásadní pro naši schopnost popisovat okolní svět, formulovat a sdělovat znalosti, třídit informace a podobně. Pojmem fuzzy množina ukázal, jak tento zásadní nedostatek odstranit. Z hlediska vícehodnotové logiky však také ukázal přirozený způsob, jak interpretovat mezilehlé pravdivostní hodnoty, např. 0.8, totiž jako pravdivostní hodnoty výroků, které obsahují vágní výrazy, tedy výroků jako „Venkovní teplota je *velmi nízká*“, „Tento pacient má *vysoký* krevní tlak“, „Bude-li inflace *vysoká*, úspory se *značně sníží*“, a podobně. Byť nám dnes připadá, že tato interpretace se přímo nabízí a nelze ji přehlédnout, v literatuře o vícehodnotové logice byla před Zadehovým člá-

Petr Hájek

nkem neznámá. Zadehův objev má atributy významného objevu: řeší problém, který dostupnými metodami řešitelný nebyl; jde o problém základní, tj. problém, který se promítá do celé řady oblastí (matematika, informatika, inženýrské obory, přírodní i společenské vědy); řeší ho koncepčně novým, ale jednoduchým pojmem, který je v článku popsán spolu se základními pravidly, jak s ním pracovat.

Zadehův pojem fuzzy množiny a obecněji pak fuzzy logika jako alternativa zobecňující logiku klasickou nepochybně představuje – řečeno termínem Thomase Kuhna – nové paradigma. Od Kuhna víme, že nové paradigma se prosazuje obtížně a je mu kladen odpor. Fuzzy logika v tomto směru nebyla výjimkou. Pro příklad uveďme, co řekl o fuzzy logice ještě v roce 1975 William Kahan, významný vědec a profesor na univerzitě v Berkeley: „Fuzzy logika je chybná, chybná a zničující. … Co potřebujeme, je více, ne méně logického myšlení. Nebezpečí fuzzy teorie je v tom, že povzbudí právě ten typ nepřesného myšlení, který nám způsobil tolik problémů.“

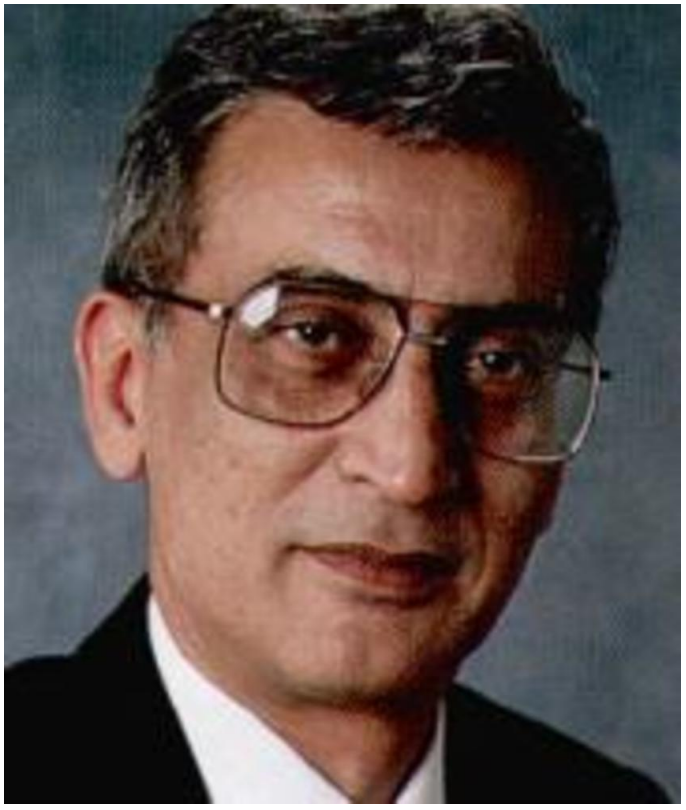
Více a více lidí však začalo chápat, že fuzzy logika má velký potenciál, že je životaschopnou alternativou logiky klasické (a že v žádném případě není nějakou defektní logikou potlačující logické myšlení, jak tvrdil například Kahan) a že – a to především – umožňuje řešit důležité problémy. Poměrně rychle se proto začal rozvíjet výzkum fuzzy logiky, který nakonec vedl k praktickým realizacím i k výrazným komerčním úspěchům.

Teoretické základy fuzzy logiky

Výzkum ve fuzzy logice, jak teoretický, tak aplikačně zaměřený, se začal významně rozvíjet od 70. let 20. století. Teoretický se přitom zabýval zejména teorií fuzzy množin a rozvojem různých oblastí matematiky, například topologie, pravděpodobnosti nebo algebry, z pohledu fuzzy množin. Aplikačně zaměřený výzkum se věnoval zejména použití fuzzy množin v různých oblastech, ve kterých hraje důležitou roli přirozený jazyk – například v rozhodování, řízení, rozpoznávání vzorů nebo klasifikaci a shlukování – a vyvinuté metody byly aplikovány pro řešení různých problémů. Významným podnětem pro rozvoj aplikačně zaměřeného výzkumu se staly první komerčně úspěšné aplikace fuzzy logiky, o kterých se zmíníme níže.

Rozvoj výzkumu byl přirozeně doprovázen rozvojem související infrastruktury: v roce 1978 vznikl první časopis věnovaný fuzzy logice – *Fuzzy Sets and Systems* –, vznikly první organizace – v roce 1982 americká NAFIPS (North American Fuzzy Information Processing Society) a v roce 1984 mezinárodní IFSA (International Fuzzy Systems Association) – a začaly se pořádat konference věnované fuzzy logice, jmenujme alespoň *IFSA World Congress* a *FUZZ-IEEE*.

Z hlediska samotné logiky, která pochopitelně tvoří ten nejníže položený základ a skutečné jádro, je třeba zmínit, že první zásadní práce, *The logic of inexact concepts* amerického informatika a matematika Josepha Goguena, se objevila už v roce 1968. Tato, z dnešního pohledu průkopnická a velmi vlivná práce, byla následována několika dalšími v 70. a 80. letech. Intenzívní rozvoj fuzzy logiky z pohledu moderní logiky ale začal až v 90. letech a probíhá dodnes. Nejvlivnější osobností na tomto poli byl Petr Hájek, český logik světového formátu. V této souvislosti je třeba také zmínit další jména českých logiků, kteří se podíleli na rozvoji fuzzy logiky – před Hájkem zejména Jan Pavelka a jeho průkopnická koncepce vycházející z Goguena, dále pak Vilém Novák, který Pavelkovu práci dále rozvinul.



Ebrahim Mamdani

Zatímco v prvních dvou dekádách se teoretický výzkum zabýval zejména tím, jak rozvinout různé oblasti matematiky, informatiky, inženýrství a dalších oblastí za předpokladu, že – poněkud nepřesně, ale stručně a výstižně řečeno – množiny jsou nahrazeny fuzzy množinami, od 90. let se těžiště začalo přesouvat k otázkám týkajícím se samotných základů fuzzy logiky a na ní založené matematiky. Ty zahrnují problémy logiky zmíněné v předchozím odstavci, ale i další otázky – pro příklad jmenujme otázku, zda a jak je možné fuzzy množiny budovat axiomaticky nebo jak nahlížet fuzzy logiku a fuzzy množiny z hlediska teorie kategorií. Jisté odpovědi na takové otázky už existují, obecně lze ale říci, že v této oblasti jsme stále spíše na začátku a že přes řadu dosažených výsledků, z nichž mnohé jsou značně netriviální, je výzkum teoretických základů stále otevřenou oblastí.

Aplikace fuzzy logiky

Komerčně jednoznačně nejúspěšnější aplikací fuzzy logiky jsou tzv. pravidlové fuzzy systémy a na nich založené fuzzy regulátory. Na ně se v této části zaměříme. Vycházejí z faktu, že v mnoha případech je člověk – zkušený operátor – schopen řídit i systém, který je značně složitý a metodami klasické teorie řízení nezvládnutelný. Na rozdíl od modelů klasické teorie řízení, které potřebují znát fyzikální model řízeného systému, se zkušený operátor řídí „algoritmem“, který je schopen popsat v přirozeném jazyce a který fyzikální model znát nepotřebuje. Tak například ví, že jestliže je teplota v místnosti velmi vysoká a otáčky větráku nízké, pak je třeba otáčky výrazně zvýšit. Zeptáme-li se ho tedy, jak vlastně systém – v tomto jednoduchém příkladě tedy větrání místnosti – řídí, řekne nám že používá několik pravidel tvaru "jestliže–pak". Tato pravidla, z nichž jedno jsme právě viděli, téměř vždy obsahují vágní výrazy („velmi vysoká teplota“ apod.) a operátor je na základě nich schopen logicky odvodit správný akční zásah, tj. správně nastavit rychlost větráku. To je ovšem živnou půdou pro fuzzy logiku, která umí vágní výrazy matematicky reprezentovat, pracovat s nimi a nakonec provádět logické úsudky, které simulují operátorovo uvažování. Výsledný systém, tzv. fuzzy regulátor, tedy v tomto smyslu napodobuje operátorovo rozhodování, aniž by bylo nutné znát fyzikální model řízeného systému. Tento poznatek je obsahem revoluční práce Ebrahima Mamdaniho a jeho studenta Sedraka Assiliana z roku 1975.

Fuzzy regulátory byly poprvé komerčně použity pro řízení pece v dánské společnosti F. L. Smidth & Company v roce 1980. Obrovský rozmach, označovaný jako „fuzzy boom“, však zaznamenaly koncem 80. a začátkem 90. let v Japonsku. Fuzzy regulátory tam byly například nasazeny k řízení metra v městě Sendai. Vlaky se tak – bez zásahů lidského operátora, plně řízené fuzzy logikou – rozjížděly i zastavovaly plynuleji, byly schopné přesně zastavit na určeném místě a dokonce spotřebovávaly o 10% méně energie. Tento úspěch později vedl k zavedení podobného systému i v Tokiu. Zejména se však fuzzy regulátory prosadily v nejrozmanitějších výrobcích na trhu se spotřební elektronikou. V obchodech se začaly ve velkém prodávat fuzzy logikou řízené fotoaparáty, pračky, vysavače, vařiče rýže a mnohé další výrobky. Podle záznamů japonského ministerstva průmyslu a obchodu činil v roce 1991 obrat japonského trhu s výrobky řízenými fuzzy logikou (70% z toho byla spotřební elektronika) cca 2 miliardy amerických dolarů, což odpovídá neuvěřitelnému 1% tehdejšího celosvětového trhu s počítačovými technologiemi.

„Fuzzy boom“ je sice již minulostí, nicméně fuzzy logika je dnes rutinně používána v mnoha oblastech.

Metro v japonském Sendai řízené fuzzy logikou

I když o tom možná nevíme, setkáváme se s ní denně i na českém trhu – je například v pračkách nebo v automatických převodovkách koncernu Volkswagen. Japonská firma Omron, jejíž měřiče tlaku jsou běžně k dostání i u nás, v roce 2013 oznámila, že prodala již 120 miliónů měřičů fungujících na bázi fuzzy logiky.

Fuzzy logika na naší katedře

Fuzzy logika je na Katedře informatiky Univerzity Palackého jedním ze stěžejních výzkumných směrů. Jako výzkumné téma jsem fuzzy logiku přinesl v roce 2001, kdy jsem na katedru nastoupil. Významnou roli sehrála i skutečnost, že jsem již v roce 2002 u nakladatelství Kluwer v New Yorku publikoval dnes ve světě poměrně široce používanou monografii *Fuzzy Relational Systems: Foundations and Principles*. Fuzzy logika se stala zcela nebo z velké části náplní práce mých doktorských studentů, z nichž se dnes fuzzy logikou většina v té či oné formě nadále zabývá. Je to v prvé řadě Vilém Vychodil, můj první doktorand, který se dnes věnuje zejména roli fuzzy logiky v relačních databázových systémech pracujících s podobností, ale i některým obecným otázkám fuzzy logiky a který je dnes na poli fuzzy logiky mezinárodně známým přispěvatelem. Dále jsou to Jan Outrata, Eduard Bartl, Jan Konečný a Petr Osička. Fuzzy logikou se také zabývá Michal Krupka a jeho studenti. Těžištěm výzkumu většiny členů katedry jsou různé aspekty teoretických základů fuzzy logiky. Pro výzkum jsme získali podporu řady grantů zejména Grantové agentury České republiky a výsledky publikovali ve významných mezinárodních časopisech v oblasti informatiky, logiky a matematiky. Podíleli jsme se však také na řešení několika aplikačně zaměřených projektů. Jedním z nich byl vývoj systému pro automatické dávkování relaxantů během celkové anestezie, na kterém jsem se podílel jako spoluřešitel. Projekt byl financován Ministerstvem zdravotnictví ČR a jeho řešitelem byl Milan Adamus, dnešní přednosta Klinika anesteziologie, resuscitace a intenzivní medicíny Fakultní nemocnice Olomouc. Systém, který používal fuzzy regulátor, byl pak používán během dlouhotrvajících operací mozku v olomoucké fakultní nemocnici.

Perspektivy

Fuzzy logika je po padesáti letech své existence stále živým oborem. Některé její oblasti mají již kanonickou podobu a jsou dnes standardní náplní učebnic. Jiné, a mezi nimi je mnoho z těch, které přímo či velmi úzce souvisí s informatikou, jsou stále otevřené a volají po řešení důležitých problémů. Dobře to ilustruje skutečnost, že prestižní Gödelova cena byla v roce 2014 udělena Ronaldu Faginovi, Amnonu Lotemovi a Monimu Naorovi právě za práci o optimální agregaci pravdivostních stupňů v jistém obecném databázovém problému.

Lotfi Zadeh popsal letos v článku *Fuzzy logic—a personal perspective* svou představu o budoucnosti fuzzy logiky takto: „Věda je v rozhodujících aspektech založena na klasické, aristotelovské, dvouhodnotové logice. Ve vědě je binarizace pravidlem spíše než výjimkou. V lidském myšlení je tomu naopak. Jedním z hlavních přínosů fuzzy logiky je to, že poskytla základ pro dalekosáhlou změnu – v téměř všech oblastech vědy – od binarismu k pluralismu, od černého a bílého k odstínům šedi. Tento posun se bude v následujících letech nejspíš zrychlovat a vliv fuzzy logiky se nejspíš stane ještě viditelnějším a významnějším. Je pravděpodobné, že ve vědě – stejně jako ve fuzzy logice – vše bude nebo bude moci být otázkou míry. To je to, co vidím ve své křišťálové kouli.“

Jednoduché nemožným a nemožné jednoduchým v datových typech

Michal Krupka

V teorii datových typů se typem rozumí jakákoli množina hodnot. Datový typ může být nekonečný, stejně jako jeho prvky. U takových typů vznikají nečekané obtíže už na základní úrovni, například s porovnáváním hodnot. Ukazují se také principiální meze schopností počítačů, například nemožnost naprogramovat nespojitou matematickou funkci. Jiné, zdánlivě neřešitelné problémy lze naopak vyřešit velmi jednoduchým programem: například průchod nekonečně velké množiny v konečném čase.

Programátoři vědí, že výpočty s reálnými čísly pomocí počítačů nejsou přesné. Konečná paměť počítače nám nedovolí ukládat reálná čísla úplně, s celým jejich nekonečným desetinným rozvojem. Proto bývají reálná čísla obvykle nahrazována čísly racionálními, která je reprezentují pouze s určitou přesností. (To platí například pro hodnoty typu `float` nebo `double` v C, Javě a C#.) Výpočty s reálnými čísly pak vyžadují velkou opatrnost, protože takto vzniklé chyby mají tendenci se při používání operací a funkcí zvětšovat, a to někdy velmi nepříjemně. Kolem problému přesnosti výpočtů s reálnými čísly vznikla celá věda.

Je ovšem možný i jiný přístup. Existuje způsob, jak provádět s reálnými čísly výpočty s libovolnou přesností. Stačí pouze začít se zdánlivě absurdním předpokladem, že paměť počítače je schopna pojmout nekonečně velká data. Další úvahy pak ještě vyžadují znalosti některých pokročilých matematických disciplín, jakou je například topologie.

K tomu zdánlivě absurdnímu předpokladu: matematici někdy rozlišují mezi *potenciálním* a *aktuálním* nekonečnem. Zatímco aktuální nekonečno znamená nekonečně velkou kvantitu, kterou chápeme jako celek (například nekonečně velkou množinu), potenciální nekonečno vyjadřuje pouze *možnost* nějakou *konečnou* kvantitu libovolně zvětšovat.

Představme si tedy, že máme reálné číslo, u něhož jsme schopni vypočítat libovolnou cifru desetinného rozvoje. To platí pro čísla e, π, $\sqrt{2}$, pro která jsou příslušné algoritmy známy, a pro mnoho dalších, která můžeme potřebovat. O takovém čísle lze říci, že je můžeme (s celým nekonečným desetinným rozvojem) uložit do počítače, protože tam můžeme uložit příslušný algoritmus.

Reprezentace čísel funkcemi

Reálná čísla tedy můžeme nahradit *funkcemi*, které pro libovolnou kladnou celočíselnou hodnotu vrátí příslušnou cifru v desetinném rozvoji daného čísla. Pro jednoduchost použijeme místo desetinného rozvoj dvojkový, takže funkce reprezentující čísla bude vracet pouze hodnoty 0 a 1. Pro další zjednodušení budeme

hovořit pouze o reálných číslech z intervalu $\langle 0; 1\rangle$, čímž se zbavíme nutnosti řešit část zápisu před desetinnou (vlastně dvojkovou) čárkou; všechna čísla z intervalu $\langle 0; 1\rangle$ lze totiž zapsat tak, že v zápise bude před čárkou nula.

Příklady si budeme ukazovat v programovacím jazyce Haskell, který je pro tuto oblast vhodný. Čtenář nebude mít problém s orientací ve zdrojovém kódu ani pokud s tímto jazykem není obeznámen. Případné zvláštnosti Haskellu vždy na místě vysvětlíme. Všechny příklady by měly být přepsatelné do jiného jazyka.

Kvůli stručnosti se dopustíme drobných programátorských prohřešků. Nebudeme také definovat potřebné datové typy. (Proto například u funkcí, které mají vracet pouze nulu nebo jedničku, to nebudeme vynucovat typovou deklarací.)

Číslo nula odpovídá funkci vracející samé nuly (je totiž $0 = 0{,}00\overline{0}$), jednička funkci vracející samé jedničky. (Ve dvojkové soustavě platí $0{,}11\overline{1} = 1$ podobně jako v desítkové $0{,}99\overline{9} = 1$.) V Haskellu je možné tyto funkce definovat takto:

```
zero n = 0
one n = 1
```

(na levé straně od rovnítka je název funkce s parametrem, na pravé hodnota).

Dále například funkce, která pro číslo n vrací nulu, pokud je n liché, a jedničku, pokud je sudé, reprezentuje číslo $0{,}0\overline{10}\overline{1}$, které je (ve dvojkové soustavě) rovno číslu $1/3$. Snadno lze napsat i funkci reprezentující iracionální číslo. Stačí, aby číslo nemělo periodický rozvoj (to nebude mít například u funkce, která vrací jedničku, pokud je n prvočíslo, a jinak vrací nulu).

Některá čísla mohou být zapsána dvěma různými způsoby. Například číslo $1/2$ má tyto dva různé zápisy: $0{,}1\overline{0}$ a $0{,}0\overline{1}$. Zápisy odpovídají těmto dvěma funkcím v Haskellu, které tedy obě reprezentují totéž číslo:

```
oneHalf 1 = 1
oneHalf n = 0

oneHalf2 1 = 0
oneHalf2 n = 1
```

(zde používáme tzv. *pattern matching*; první řádek definice vždy říká hodnotu funkce pro parametr roven jedné, druhý pak pro ostatní hodnoty). Dvojkový zápis čísla tedy nemusí být jednoznačný. To je třeba mít stále na paměti, například v dalším odstavci, když budeme mluvit o neexistenci netriviálního predikátu.

Napišme si ještě dvě pomocné funkce pro práci s posloupnostmi:

```
rest x = \n -> x(n+1)

prepend val x 1 = val
prepend val x n = x(n - 1)
```

Funkce rest vrací k posloupnosti x posloupnost začínající jejím druhým členem. Protože posloupnosti (a potažmo i čísla) reprezentujeme funkcemi, vrací funkce pro každé x novou (anonymní) funkci. Tu zapisujeme haskellovským způsobem pro zápis lambda výrazu: \n -> x(n+1) značí funkci, která pro parametr n vrátí hodnotu funkce x v parametru n+1.

Funkce prepend naopak připojí k dané posloupnosti (parametr x) nový prvek (parametr val).

Kromě intervalu $\langle 0; 1\rangle$ budeme v příkladech používat i *Cantorův prostor*, což je množina všech posloupností nul a jedniček. Tyto dvě množiny jsou velmi příbuzné, v našich příkladech budeme prvky obou reprezentovat stejně – posloupnostmi nul a jedniček. Rozdíl mezi čísly a posloupnostmi je v tom, že jedno číslo lze někdy reprezentovat dvěma různými posloupnostmi, zatímco u posloupností samotných je reprezentace jednoznačná.

Reálná čísla z intervalu $\langle 0; 1\rangle$ lze v počítači modelovat i jinou datovou strukturou než posloupnostmi nul a jedniček. Náš přístup ve skutečnosti není příliš výhodný. V tomto článku jsme se k němu uchýlili, protože je snadno pochopitelný.

Interval $\langle 0; 1\rangle$ a Cantorův prostor jsou příklady *datových typů*, což je v teorii typů libovolná (i nekonečná) množina hodnot reprezentovatelných v počítači.

Přísně vzato, ne všechna reálná čísla z intervalu $\langle 0; 1\rangle$ (a ne všechny posloupnosti nul a jedniček) lze v paměti počítače reprezentovat. Tuto okolnost zcela pomineme. Bude nám stačit, že s takovými čísly se těžko někdy setkáme (například čísla e, π, $\sqrt{2}$, jak už bylo řečeno, to nejsou).

Predikáty

Predikát je funkce, která vrací hodnotu *Ano* nebo *Ne* (v Haskellu **True** nebo **False**). Jedná se o základní typ funkcí, které můžeme pro datový typ naprogramovat. Predikáty jsou funkce, které rozhodují, zda daná hodnota má nějakou vlastnost, nebo více hodnot mezi sebou nějak porovnávají.

Nejprve budeme hovořit o tzv. *úplných predikátech*, které jsou definovány pro všechny hodnoty datového typu. Později se dostaneme k neúplným predikátům.

Při úvahách nad predikáty na intervalu $\langle 0; 1\rangle$ narazíme na první překvapení. Zamysleme se nad tím, jak naprogramovat funkci, která zjistí, zda se dvě reálná čísla rovnají. Pokud pomineme možnost, že by zápis jednoho z čísel obsahoval posloupnost samých jedniček, která by věc ještě zkomplikovala, jde nám o to, zda jsou posloupnosti čísla reprezentující stejné. To ale, pochopitelně, nelze v konečném čase zjistit. Například čísla

$$a = 0{,}100111101011\ldots$$
$$b = 0{,}100111101011\ldots$$

mají prvních 12 cifer stejných, což ale neznamená, že se nebudou lišit ve třínácté, nebo třeba tisící první cifře. Musíme tedy udělat tento závěr: *nelze napsat program, který v konečném čase rozhodne, zda se dvě reálná čísla rovnají*. Odborně řečeno: na datovém typu $\langle 0; 1\rangle$ *není rozhodnutelná rovnost*.

Situace je ovšem ještě dramatičtější. Pro reálná čísla nelze korektně naprogramovat *vůbec žádný netriviální predikát*! Důvod, proč tomu tak je, naznačíme později. Jediné dva predikáty s jedním parametrem, které lze korektně naprogramovat pro čísla z intervalu $\langle 0; 1\rangle$, jsou tyto:

```
pred1 x = True
pred2 x = False
```

Podobný závěr platí i pro predikáty s více parametry. Mezi ně patří (kromě výše zmíněného predikátu rovnosti) i standardní relace na porovnání čísel, jako jsou relace „menší“ a „menší nebo rovno“. Žádná z nich není rozhodnutelná, žádnou z nich nelze korektně naprogramovat.

Poznamenejme, že náš druhý datový typ, Cantorův prostor, tak velkým nedostatkem predikátů netrpí. Jistě, zjistit, zda jsou dvě hodnoty stejné, nelze ani zde. Rovnost je tedy nerozhodnutelná i v Cantorově prostoru. Jiné predikáty ale naprogramovat můžeme. Následující například zjišťuje, zda posloupnost x má na osmém nebo desátém místě jedničku:

```
pred3 x = x(8) == 1 || x(10) == 1
```

(operátor || počítá logickou operaci *nebo*). Necháme na čtenáři, aby si vyjasnil, proč pred3 nelze používat na reálná čísla.

Nedostatek predikátů nás nutí oslabit požadavky, které na ně klademe. Proto se v teorii datových typů často používají *částečné predikáty*. To jsou funkce, které se místo vrácení hodnoty **False** mohou dostat do nekonečné smyčky. Jako částečný predikát lze napsat funkci, která zjistí, zda se dvě čísla nerovnají, i funkci, která zjistí, zda je jedno číslo menší než druhé (opět pro jednoduchost neuvažujeme, že by zápis jednoho z čísel obsahoval od určité pozice samé jedničky; čtenář si může definice vylepšit, aby počítaly i s touto možností).

```
notEqual x y =
  x(1) /= y(1) || notEqual (rest x) (rest y)

less x y =
   x(1) <= y(1) && (x(1) < y(1) || less (rest x) (rest y))
```

Obě funkce se dostanou do nekonečné smyčky, pokud jsou zavolány se dvěma stejnými hodnotami. Jinak vracejí správný výsledek.

Množiny a matematické funkce

Predikáty a částečné predikáty lze používat na *rozpoznávání množin* v datovém typu. Množina je rozpoznána daným (částečným) predikátem, pokud je tvořena právě hodnotami, pro které predikát vrací **True**. Jinak řečeno, je to množina prvků splňujících podmínku danou predikátem.

Například množina všech čísel větších než $1/2$ je rozpoznána (částečným) predikátem

```
pred4 x = less oneHalf x
```

(číslo oneHalf je definováno výše). Pro zajímavost poznamenejme, že Haskell umí tzv. *currying*, takže definice predikátu pred4 může vypadat i takto:

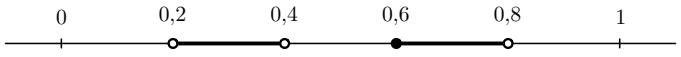
```
pred4 = less oneHalf
```

Z toho, co jsme si už řekli, vyplývá, že ne každá množina v datovém typu musí být rozpoznatelná. Například jednoprvkové množiny rozpoznatelné nejsou. Proč? Brání tomu nerozhodnutelnost rovnosti. Kdybychom třeba chtěli zjistit, zda prvek x je prvkem množiny $\{0\}$ (jednoprvková množina obsahující nulu), museli bychom zjistit, zda se rovná nule, což, jak víme, nejde.

Rozpoznatelné množiny v libovolném datovém typu (tedy nejen v intervalu $\langle 0;1\rangle$ a Cantorově prostoru) jsou množiny, které jsme schopni definovat programem. Fakt, že ne každá množina je rozpoznatelná, ukazuje na omezené možnosti počítačů.

Lze rozpoznatelné množiny nějak popsat? Odpověď je kladná a přichází z oblasti matematiky, která je často v záležitostech informatiky a programování považována za okrajovou: z topologie. Ukazuje se totiž, že rozpoznatelné množiny jsou právě ty množiny, kterými se topologie zabývá. Jsou to tak zvané *otevřené množiny*.

V oboru reálných čísel jsou otevřené ty množiny, které neobsahují své hraniční body. Co je tím myšleno, vysvětlíme pomocí příkladu. Na obrázku vidíme dva intervaly: otevřený interval $(0{,}2;0{,}4)$ a zleva uzavřený a zprava otevřený interval $\langle 0{,}6;0{,}8)$.



Hraničními body prvního intervalu jsou čísla $0{,}2$ a $0{,}4$. Interval ani jedno z nich neobsahuje, je tedy otevřenou množinou. To znamená, že je i rozpoznatelný, což by neměl být problém ověřit napsáním funkce, která vrací **True** právě pro prvky intervalu $(0{,}2;0{,}4)$. Necháme na čtenáři, aby se o napsání takové funkce pokusil. Jen připomínáme, že pro ostatní vstupy má funkce povoleno dostat se do nekonečné smyčky.

Hraničními body intervalu $\langle 0{,}6;0{,}8)$ jsou čísla $0{,}6$ a $0{,}8$. Jelikož první z nich v intervalu leží, není tento interval rozpoznatelný. Neexistuje funkce, která by vracela **True** právě pro čísla z intervalu $\langle 0{,}6;0{,}8)$.

Nemožnost rozpoznat přesně interval $\langle 0{,}6;0{,}8)$ je důsledkem naší neschopnosti rozpoznat přesně jednotlivé číslo (v tomto případě $0{,}6$). Pro libovolné číslo a menší než $0{,}6$ ovšem můžeme napsat predikát rozpoznávající otevřený interval $(a;0{,}8)$. Tak se můžeme k intervalu $\langle 0{,}6;0{,}8)$ libovolně přiblížit. Záleží jen na nás, s jakou přesností se spokojíme, vždy se ale nějaké chyby dopustíme.

Otevřené množiny má i Cantorův prostor. Z našeho pohledu jsou to přesně ty množiny, které jsou rozpoznatelné predikáty. Necháme na čtenáři, aby odhadl, které to jsou.

Matematické funkce (počínaje sčítáním, odčítáním, násobením a dělením a pokračuje složitějšími funkcemi jako např. polynomy, exponenciálními, logaritmickými a goniometrickými funkcemi) lze všechny pro datový typ $\langle 0;1\rangle$ naprogramovat. Příklady si zde ukazovat nebudeme, protože naše jednoduchá reprezentace čísel posloupnostmi nul a jedniček není pro tento účel dostatečná. Poznamenejme pouze, že funkce

```
div2 x = prepend 0 x
```

realizuje dělení dvojkou.

Matematické funkce, které lze naprogramovat, se nazývají *vyčíslitelné*. Základní výsledek teorie datových typů říká, že *všechny vyčíslitelné funkce jsou spojité*. Spojitost matematické funkce je základní topologický pojem, studenti vysoké školy se s ním obvykle seznamují v matematické analýze. Zhruba řečeno platí, že funkce je spojitá, pokud při kreslení jejího grafu nemusíme zvednout tužku z papíru.

Uvedený výsledek tedy znamená, že *nelze naprogramovat nespojitou funkci*. To je také důvod, proč na reálných číslech neexistuje netriviální úplný predikát. Pokud si hodnotu **False** vyjádříme jako číslo 0 a hodnotu **True** jako číslo 1, vidíme, že nelze bez zvednutí tužky z papíru nakreslit funkci, která by nabývala pouze těchto dvou hodnot.

Průchod nekonečnou množinou

Poznatky, které jsme se zatím dozvěděli, lze shrnout takto: Na počítači je možné pracovat s reálnými čísly s libovolnou přesností, když se čísla vhodně reprezentují. Cenou za přesnost výpočtů je neřešitelnost některých základních, zdánlivě triviálních problémů: problém rovnosti čísel i posloupností je nerozhodnutelný, nelze naprogramovat netriviální úplný predikát pro reálná čísla ani (obecněji) žádnou nespojitou funkci.

Z obecnějšího pohledu můžeme říci, že na počítačích lze pracovat i s datovými typy, jejichž hodnoty samy mají nekonečně mnoho prvků. Kromě reálných čísel je příkladem takového datového typu množina všech nekonečných posloupností nul a jedniček. U takových typů můžeme zkoumat množiny, které lze rozpoznat naprogramovatelnými funkcemi (predikáty), a zabývat se vyčíslitelnými funkcemi.

Na závěr se podíváme na jeden překvapivý výsledek, poprvé uvedený v roce 1990 Ulrichem Bergerem v jeho doktorské práci.

Představme si, že jsme naprogramovali (úplný) predikát p na Cantorově prostoru. Následující funkce forAll je predikát, který má hodnotu **True**, když je predikát p splněný *pro každou posloupnost z Cantorova prostoru*. Jinak vrací **False**. Pomocná funkce forSome je predikát, který vrací **True**, právě když existuje posloupnost, která splňuje zadaný predikát, a pomocná funkce find tuto posloupnost vrací jako svůj výsledek. Funkce se vzájemně volají.

Pro pochopení funkcí je potřeba vědět, že na rozdíl od běžných programovacích jazyků Haskell používá *líné vyhodnocování*, tj. nevyhodnocuje argumenty funkcí, dokud to není nutné (podobně se běžné programovací jazyky chovají při vyhodnocování logických operací).

```
forSome p = p(find(p))

find  p =
  if  forSome(\x -> p(prepend 0 x))
      then  prepend 0 (find(\x -> p(prepend 0 x)))
      else  prepend 1 (find(\x -> p(prepend 1 x)))

forAll  p = not(forSome (\x -> not(p x)))
```

Zkouška správnosti funkce forAll pro dříve definované predikáty pred1, pred2 a pred3:

```
*Test> forAll pred1
True
*Test> forAll pred2
False
*Test> forAll pred3
False
```

Napsali jsme funkci, která *v konečném čase projde nekonečně velkou množinu* (celý Cantorův prostor všech nekonečných posloupností nul a jedniček) *a zjistí, zda všechny její prvky splňují zadanou podmínku*.

Kromě funkce forAll existují i další funkce řešící zdánlivě neřešitelné problémy. Ukazuje se například, že lze napsat predikát, který rozhodne, zda jsou dva predikáty na Cantorově prostoru shodné. Na množině (datovém typu) všech predikátů na Cantorově prostoru je tedy rozhodnutelná rovnost.

Hlubší topologické úvahy ukazují, že podobné funkce lze napsat nejen pro Cantorův prostor, ale i pro jiné nekonečné datové typy. Jsou to typy, které jsou, podle topologické terminologie, *kompaktními prostory*. Takovým typem jsou například mnohé podmnožiny Cantorova prostoru, takže lze napsat funkci, která zjistí, zda je daný predikát splněn pro všechny posloupnosti nul a jedniček, vyhovující nějaké dodatečné podmínce.

Pokud místo úplných predikátů použijeme částečné, dosáhneme stejných výsledků i na intervalu $\langle 0; 1\rangle$ (víme, že na tomto intervalu neexistují netriviální úplné predikáty, takže pracovat jen s nimi by nemělo smysl).

Důvod, proč funkce forAll a jí podobné vždy skončí, není možné v tomto článku rozebírat. Pro naši konkrétní funkci forAll jej můžeme odhadnout, pro obecný případ, jak už bylo řečeno, zdůvodnění vyžaduje hlubší topologické úvahy. Podstatným předpokladem pro správný výsledek funkce forAll je, že predikát p, který je jejím parametrem, je někým naprogramovanou, a tedy vyčíslitelnou funkcí. Kdyby vznikl jinak než naprogramováním (byl by třeba výsledkem nějakého fyzikálního procesu), funkce forAll by mohla selhat.

Otázka, zda lze jiným způsobem než naprogramováním vytvořit nevyčíslitelnou funkci, zůstává otevřená. Kdyby byl celý vesmír jeden veliký počítač, určitě by to nešlo.

Hádanka

Petr Krajča

V drtivé většině případů platí, že hodnota se rovná sama sobě, tj. x == x je pravdivý výraz. Víte ale pro jaký datový typ a hodnotu to neplatí? Pro vetší představivost přikládáme kód v jazyce C, avšak hádanka platí i pro další běžné jazyky jako Java, C# nebo třeba JavaScript.

```
#include <stdio.h>
#include <math.h>


int main()
{
        <datovy typ> x = <hodnota>;
        if (x == x)
                printf("Tento text se nikdy nezobrazi\n");
        return 0;
}
```

Správné řešení hádanky naleznete na webové stránce magazínu.

Redakce:
Petr Krajča, Petr Osička
Katedra informatiky PřF UP
17. listopadu 12
771 46 Olomouc

web: www.inf.upol.cz/magazin
email: magazin@inf.upol.cz
atom: http://www.inf.upol.cz/atom/magazin
telefon: 585634701
GPS: 49.591870, 17.262336