Logika

HASHIM HABIBALLA

Přírodovědecká fakulta OU, Ostrava - Gymnázium Olgy Havlové, Ostrava-Poruba -

Fakulta prírodných vied UKF, Nitra

Logika je oproti jiným disciplínám teoretické informatiky vědou velmi starou a její kořeny lze najít již ve starověku. Tehdy byla spjata spíše s filozofickými otázkami než s rodící se matematikou. Přesto otázky, které byly v té době aktuální, jsou v mnohém stejné jako je uplatnění logiky v informatice. Na logiku je možné se dívat v kontextu mnoha věd a výsledný pohled může být velmi odlišný. Touto vědou se zabývají filozofové, právníci, matematici i informatici a jejich zájem a cíl bývá velmi odlišný. Čtenář může sám posoudit, jak odlišné mohou tyto pohledy být - stačí si pročíst například filozoficky a informaticky orientovanou knihu o logice. Přesto lze najít společnou snahu o **modelování lidského úsudku**. I když matematik logiku spíše používá k formulaci a dokazování vlastností matematických objektů, pro informatika je logika nejen nástrojem, ale i plnoprávnou disciplínou zkoumanou v rámci teoretické informatiky.

Přístupy při modelování úsudku založené na logice patří do rodiny symbolických přístupů, existují pak také tzv. konekcionistické přístupy (např. umělé neuronové sítě), ale těmito přístupy se nebudeme zabývat (viz [8]). Tyto přístupy se často inspirují biologickými procesy a snaží se je modelovat matematicky. Často se objevují také velmi úspěšné kombinace obou přístupů.

Díky své bohaté historii je samozřejmě logika velmi širokým oborem a v tomto článku bychom se chtěli zabývat několika důležitými otázkami z informatického pohledu.

1. Reprezentace znalostí klasickou logikou - výroková a predikátová logika.
2. Automatizovaná dedukce - formální systémy pro dokazování vět.
3. Vícehodnotová logika - možnosti modelování neurčitosti pomocí fuzzy logiky.

Chceme opět jako v předchozích článcích zdůraznit, že nám nejde o vytvoření rigorózní monografie, ale o text, který čtenáři přinese chuť do dalšího studia i do výuky logiky. Doporučujeme velmi inspirativní a přitom vysoce odbornou informatickou monografii [4].

1 Reprezentace znalostí klasickou logikou

Klasickou logikou myslíme formalismus, který je v různých podobách znám již stovky let. Lidé jsou schopni komunikovat a formulovat své myšlenky v přirozeném jazyce a dále je zpracovávat a **dedukovat závěry**. Právě přirozený jazyk je však poměrně složitý pro stroje, které by měly simulovat tento způsob reprezentace znalostí. Proto logika nabízí různé úrovně zjednodušení formulace znalostí do co nejmenšího množství exaktně definovaných symbolů. Druhou stránkou je motivace, abychom z těchto statických znalostí reprezentovaných symboly dokázali také odvozovat závěry. K tomu slouží různé symbolické metody, z nichž některé si popíšeme v tomto článku. Proveďme analogii s právními předpisy. Samotné zapsané zákony jsou sice velice užitečné, ale k čemu by nám byly, pokud by nebylo možné je interpretovat a zjišťovat, zda se například některý subjekt nedopustil porušení zákona. To můžeme provést jedině tak, že dedukujeme závěr (zda někdo jedná protizákonně nebo ne) z předpokladů (fakta popisující situaci a dané zákony).

Nejjednodušším formalismem vhodným pro reprezentaci znalostí je výroková logika. U každé logiky si musíme uvědomit, že má svou **syntaxi** a **sémantiku**. Zjednodušeně můžeme říci, že syntaxe je definicí symbolů a jejich používání (bez ohledu na význam těchto symbolů). Sémantika je pak chováním těchto symbolů s ohledem na smysl daných znalostí (např. pravdivost daných tvrzení). Právě možnost oddělit syntaxi a sémantiku ,po formulaci a dokázání potřebných vztahů mezi nimi, dává logice význam v informatice a pro automatizaci úsudku.

Syntaxe výrokové logiky pracuje se symboly zastupujícími elementární výroky (např. a,b,c,...), logické konstanty (pravda a nepravda - $\top$,$\bot$) a dále je umožňuje spojovat do složitějších formulí pomocí symbolů logických **spojek** a pomocných symbolů (např. $\wedge$ konjunkce, $\rightarrow$ implikace). Sémantika pak popisuje smysl těchto použitých symbol a pojmy, které souvisejí s popisem tohoto smyslu (tzv. interpretací formulí - v případě klasické logiky to může být buď formule pravdivá nebo nepravdivá). Jednotlivým výrokům můžeme přiřadit logickou hodnotu podle toho, jak se tyto syntaktické elementy chovají v našem modelovaném případě z reality, příp. to můžeme udělat také zcela nesmyslně. V obou případech pak můžeme uvažovat jaká bude interpretace celých složených formulí. Záleží to na použitých unárních a binárních spojkách. Čtenář se zřejmě setkal se spojkami jako je negace $\neg$(opak ve smyslu pravdivosti), konjunkce $\wedge$ (současná platnost výroků), disjunkce $\vee$ (platnost alespoň jednoho z výroků), implikace $\rightarrow$ (podmínka), ekvivalence $\leftrightarrow$ (identická pravdivost výroků). Binárních spojek existuje samozřejmě více - v elektrotechnice se kupříkladu používají spojky jako je negace konjunkce (nand). Formule tedy v tomto případě platí (je interpretována jako pravdivá), pokud neplatí dva výroky současně (jinak řečeno pokud alespoň jeden neplatí). Druhým případem, který se zase vyskytuje často v reálných situacích je tzv. vylučovací nebo. V přirozené řeči často používáme výrok typu "buď ... anebo ...", který spíše odpovídá situaci, že musí platit alespoň jeden z výroků, ale zároveň nesmí platit oba současně. Nejde tedy o disjunkci, ale spojku, kterou někdy také označujeme jako negaci ekvivalence. Interpretaci logických spojek můžeme přehledně realizovat pomocí tabulky, ve které jsou vypsány všechny možné kombinace ohodnocení

pravdivosti elementárních výroků. Interpretaci pravda označíme 1 a nepravda 0. Podívejme se na příklad identického chování negace ekvivalence a vylučovacího nebo.

výrok A	výrok B	vylučovací nebo	ekvivalence	negace ekvivalence
0	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	0

V klasické logice existuje pouze konečný (a navíc velmi omezený) počet spojek. Stačí se zamyslet nad všemi možnostmi, které mohou nastat a dojdeme k tomu, že pro 2 výroky ohodnocené 4 možnostmi pravda/nepravda existuje 2^4 možností dvouhodnotové interpretace tedy 16 možných spojek.

Sémantika jazyka výrokové logiky umožňuje na základě interpretace formule definovat další důležité pojmy.

Pokusme se nyní na příkladu namodelovat pomocí výrokové logiky jednoduchou situaci. Příklad 1:

K soudu byli předvedeni tři podezřelí z loupeže - A, B a C. Při výslechu se zjistily tyto skutečnosti:
1. Do případu nebyl zapleten nikdo jiný než A, B a C.
2. A pracuje vždycky alespoň s jedním společníkem.
3. C je nevinen.

Nejprve si musíme stanovit, co jsou elementární výroky. Jelikož se vyjadřujeme k vině a nevině podezřelých, bude rozumné pomocí výroků A, B a C symbolizovat, že daný podezřelý je vinen nebo nevinen. Pak můžeme zapsat větu 1. jako složený výrok vyjadřující pomocí disjunkce, že alespoň jeden z podezřelých je vinen. Druhá věta může být popsána formulí pomocí implikace (podmínky), která říká, že je-li vinen A, pak musí být vinen také alespoň jeden z podezřelých B, C. Výsledné formule výrokové logiky jsou tedy následující:

1. $A \vee B \vee C$, 2. $A \rightarrow (B \vee C)$, 3. $\neg C$

Samotná reprezentace znalostí je pouze polovičním úspěchem při pokusu o modelování úsudku. Dalším nevyhnutelným krokem je možnost ověřovat, zda nějaké tvrzení vyplývá z daných předpokladů. K tomu si nejprve musíme definovat další pojmy sémantiky. Jde o tzv. **splnitelnost** (nesplnitelnost) a **platnost** formulí. Splnitelná formule je laicky řečeno taková, která má vůbec smysl pro určité ohodnocení výroků. Tedy taková formule musí alespoň pro nějakou kombinaci ohodnocení výroků pomocí pravda/nepravda dávat interpretaci pravda. Jinak je taková formule nesplnitelná resp. v logice se takové formuli říká kontradikce. Pokud bychom se obrátili zpátky na interpretační tabulku, pak by tabulka ve sloupci interpretace splnitelné formule musela mít alespoň v jednom řádku symbol 1.

Některé ze splnitelných formulí pak mohou být ještě na vyšším sémantickém stupni a mohou být pravdivé pro všechna možná ohodnocení. Takovým formulím se pak říká platná formule resp. v logice je zažitý pojem tautologie. Podívejme se na příklad.

Příklad 2:
Mějme platné tvrzení: "Pokud prší, beru si deštník."Jestliže namodelujeme tvrzení, že "prší"pomocí P a "beru si deštník"pomocí D, pak výsledná formule je následující:
$P \rightarrow D$
Uvažujme nyní o výroku: "Pokud si neberu deštník, neprší."Je takové tvrzení ekvivalentní prvnímu? Platí-li první tvrzení, mohu ho brát jako postulát, kterému se každý musí podřídit. Zdá se tedy logické, že když si deštník neberu a dodržuji přitom postulát, pak nemůže pršet. Mnohem formálněji bychom tento vztah mohli dokázat právě pomocí pojmu tautologie. Druhá forma tvrzení se dá zapsat jako $\neg D \rightarrow \neg P$. Pak využijeme spojky ekvivalence, která interpretuje jako pravdivé dvě formule se stejnou pravdivostní hodnotou a tedy vlastně popisuje ekvivalentní chování dvou formulí z hlediska pravdivosti. Sestrojíme tedy takovou formuli a prokážeme, že je to tautologie pomocí interpretační tabulky. Implikace je nepravdivá pouze pro případ, kdy první formule je pravdivá a druhá nepravdivá - to je v souladu s naším chápáním podmínky. Pokud je splněn předpoklad podmínky, pak závěr musí platit - jinak by nebyla formule pravdivá. V případech kdy podmínka splněna není, může i nemusí závěr platit a v obou případech je to v pořádku. Jelikož sestrojená ekvivalence obou formulí je pravdivá ve všech interpretacích, můžeme konstatovat, že jde opravdu o tautologii.

P	D	$P \rightarrow D$	$\neg D$	$\neg P$	$\neg D \rightarrow \neg P$	$(P \rightarrow D) \leftrightarrow (\neg D \rightarrow \neg P)$
0	0	1	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1
1	1	1	0	0	1	1

Nyní se již konečně dostáváme ke klíčovému pojmu tzv. **logického důsledku** množiny předpokladů. Máme-li množinu předpokladů a závěr, můžeme se ptát, zda tento závěr je logickým důsledkem (vyplývá z) předpokladů. To platí v případě, že pro každé ohodnocení výroků s interpretací pravda pro všechny formule z množiny předpokladů je pravdivá také formule reprezentující závěr. Jde tedy o chování podobné implikaci. Závěr nebude logicky vyplývat z předpokladů, jen pokud se najde takové ohodnocení výroků, kdy všechny předpoklady jsou pravdivé a závěr není pravdivý. Tato činnost, kdy odvozujeme závěry, se nazývá dedukce. Důležitým faktorem pak ještě zůstává tzv. konsistence předpokladů. Jde o to, že formulovaná vlastnost vyplývání degraduje na triviální situaci, pokud předpoklady nemají žádné ohodnocení, ve kterém by byly současně pravdivé. Takovéto množiny předpokladů jsou sporné (nekonsistentní) a z hlediska definice důsledku, je důsledkem této množiny jakákoliv formule. To samozřejmě v realitě není zcela odpovídající. Jednalo by se například o situaci, kdy určitý balík zákonů obsahuje dvě navzájem protichůdná nařízení a podle této množiny by pak jakékoliv jednání bylo v souladu s tímto

zákonem.

Příklad 3:
Pokusme se nyní jednoduše pomocí tabulky prověřit, zda z množiny předpokladů z příkladu 1. vyplývá vina či nevina některého z podezřelých. Můžeme vyloučit podezřelého C, protože jeho nevina je přímo obsažena v předpokladech a tudíž musí vyplývat z množiny předpokladů a zároveň nemůže vyplývat jeho vina, pokud není množina předpokladů sporná. Postačí tedy sestrojit tabulku pro jednotlivé předpoklady a pro formule A,$\neg A$,B,$\neg B$ ověříme, zda nesplňují vlastnost důsledku. V tabulce se vyskytuje také sloupec s hvězdičkami, který zvýrazňuje ohodnocení, kde jsou všechny předpoklady platné a tedy v těchto řádcích musíme kontrolovat interpretaci potenciálního důsledku. U závěrů pak vyznačujeme pomocí hvězdičky resp. lomítkem, zda skutečně splňuje resp. nesplňuje formule podmínku důsledku. Pokud ji splňují všechna zvýrazněná ohodnocení jde skutečně o vyvoditelný logický důsledek.

A	B	C	$A \vee B \vee C$	$A \to (B \vee C)$	$\neg C$		A		$\neg A$		B		$\neg B$	
0	0	0	0	1	1		0		1		0		1	
0	0	1	1	1	0		0		1		0		1	
0	1	0	1	1	1	*	0	/	1	*	1	*	0	/
0	1	1	1	1	0		0		1		1		0	
1	0	0	1	0	1		1		0		0		1	
1	0	1	1	1	0		1		0		0		1	
1	1	0	1	1	1	*	1	*	0	/	1	*	0	/
1	1	1	1	1	0		1		0		1		0	

Z tabulky je patrné, že jediný logický důsledek z prověřovaných závěrů je, že B je vinen. O podezřelém A nemůžeme konstatovat na základě předložených předpokladů ani jeho vinu ani nevinu. Vezmeme-li v úvahu možnost, že A je vinen, pak by musel být vinen i B. Pokud A vinen není, zároveň C vinen není dle předpokladu a někdo vinen být musí, pak padá vina na B. B je tedy vinen v každém případě a o A nás ale nic neopravňuje to tvrdit.

Na uvedeném příkladu dedukce může ilustrovat hned několik klíčových otázek a problémů, které s logikou a naším článkem souvisí.
1. Pomocí přesně definovaných postupů jsme schopni dedukovat důsledky zcela automatizovaně a to zvládnou nejen lidé, ale zejména je to vhodné pro stroje (počítače). Už námi řešený příklad nemusí být pro každého člověka jednoduchý a složitost dedukce roste jednak s počtem výroků a jednak s počtem předpokladů. Už z těchto důvodů nejsou prostředky logiky zbytečné.
2. Problém automatizované dedukce lze řešit různě efektivní metodami. V tomto článku jsme se setkali prozatím jen s tabulkovou metodou, která je sice velmi triviální a dokonce si asi čtenář dokáže představit, že by takovýto algoritmus dokázal naprogramovat, nicméně její jednoduchost je také její slabinou. Už na příkladu 2 a 3 můžeme vidět, že rozsah tabulky

roste exponenciálně vzhledem k počtu elementárních výroků. S ohledem na poznatky teorie vyčíslitelnosti a složitosti [3]víme, že exponenciální časová a prostorová složitost je pro klasické deterministické počítače prakticky nepoužitelná. Už pro 10 výrokových proměnných je možností ohodnocení přes 1000, pro 20 přes milion atd... Proto bylo a stále je vyvíjeno mnoho různorodých metod a celých formálních systémů, které jsou schopny automatizovat dedukci mnohem efektivněji.
3. Klasická výroková logika je jednou z nejjednodušších logik, což má své výhody i nevýhody. Výhodou je její transparentnost, pochopitelnost a především vlastnost rozhodnutelnosti [3]. Rozhodnutelnost ve smyslu schopnosti o dané formuli říci jednoznačně, zda je splnitelná nebo ne na základě algoritmu, je důležitou vlastností pro automatizaci dedukce. Složitější logiky tuto vlastnost mít nemusejí. Nevýhodou výrokové logiky je naopak neschopnost rozlišit objekty a vztahy mezi nimi, nemožnost kvantifikovat vztahy, pojmout proměnlivost pravdivosti v čase a podobně. V neposlední řadě je u klasických logik nevýhodou jejich "černobílé vidění světa". Myslíme tím neschopnost zachytit jinou než absolutní pravdivost nebo nepravdivost. V reálném životě je mnoho situací, které nelze popsat jednoznačně. Například to zda je člověk spokojený, lze stěží popsat jen výrokem pravdivým nebo nepravdivým. Člověk může být spokojený v určitém stupni spokojenosti. Proto se v tomto článku chceme dotknout i tématu vícehodnotových logik, které jsou nyní velmi intenzivně zkoumány.

Ještě než se budeme blíže zabývat otázkou automatizace dedukce, chtěli bychom také stručně popsat logiku predikátovou. Predikátová logika je v podstatě zobecněním logiky výrokové a dává ji schopnost pracovat nejen s elementárními výroky, ale také rozlišit objekty a jejich vztahy. Na syntaktické úrovni se zavádí pojem termu a formule. Termem může být buď symbol zastupující objekt (konstanta) nebo funkci (funktor) nebo proměnná, za kterou lze dosadit libovolný term. Klíčovým pojmem je predikát, který jako argumenty může mít termy a tím vlastně umožňuje vytvářet vazby mezi termy. Například bychom chtěli prostřednictvím predikátu zachytit vazby mezi rodiči a jejich dětmi. Zavedli bychom predikát s názvem dítě, který má dva argumenty - dítě a jeho rodiče. Samozřejmě pracujeme se symboly, takže skutečnými argumenty jsou pouze konstanty reprezentující objekty - konkrétní děti, rodiče atd. Konstanty jsou nejjednoduššími termy. Termy tedy nevyjadřují narozdíl od predikátů vazby, ale zastupují symbolicky objekty modelované reality. Takovými konstantami by mohla být například jména dětí a jejich rodičů. Tedy bychom mohli sestavit velmi jednoduchou formuli dítě(johana, lucie), která by měla pomocí predikátu vyjadřovat znalost, že mezi symbolem reprezentujícím objekty johana a lucie je vztah dítěte a rodiče. Termy mohou být však složitější a to funktory a proměnné. Funktory jsou symbolickými ekvivalenty pro nám dobře známé funkce. Funkce může mít několik argumentů (opět termů) a její interpretací je pak požadovaná hodnota. Například goniometrická funkce cos by pro argument - symbol 0 (intepretovaný číslem 0) - vracela číslo 1 ($cos(0) = 1$). Proměnné pak jsou symbolickým prvkem, do kterých mohou být dosazeny jiné termy. Jelikož interpretace termů a predikátů zaleží (narozdíl od logických spojek) na uživateli predikátové logiky, mohl by si sémantiku uživatel uzpůsobit dle svých požadavků. Mohl by tedy vytvořit absurdní intepretace, kde by například funkce s názvem cos byla interpretována pro argument - číslo 0 - třeba číslem 333 nebo zcela nečíselným objektem.

Chceme tím říci, že predikátová logika je velmi flexibilní a je potřeba mít stále na paměti rozdíl mezi syntaxí a sémantikou. Symboly mají své interpretace (sémantiku), kterou v případě logiky predikátové značně ovlivňuje ten, kdo ji používá. A i pod zcela totožnými symboly se tak může skrývat (nekonečně) mnoho různých významů.

Predikát může ze sémantického hlediska nabývat opět hodnotu pravda nebo nepravda. Jeho sémantickým operátorem je relace. Od úrovně predikátů výše se pak formule chovají jako ve výrokové logice a můžeme je tedy spojovat pomocí logických spojek. Jediným rozdílem je, že můžeme formule kvantifikovat a to buď univerzálním ($\forall x$) resp. existenčním ($\exists x$) kvantifikátorem. Ty pro zvolenou proměnnou pak zaručují, že kvantifikovaná formule bude pravdivá pro všechny resp. alespoň jeden objekt dosaditelný za proměnnou x.

Příklad 4:
Chtěli bychom pomocí predikátové logiky namodelovat situaci, kdy libovolné dítě je šťastné, pokud má otce i matku. Zavedli bychom si predikátové symboly pro vlastnosti objektů - stastny, muz a zena a dále pro vztah býti dítětem někoho - dite. Pomocí formule 1. pak můžeme vyjádřit, že pro každé dítě, ke kterému existuje objekt, jenž je ženou a zároveň dítě je dítětem tohoto objektu a zároveň platí totéž pro objekt typu muž, pak platí, že toto dítě je šťastné. Mnohem jednodušší je pak vyjádřit, které objekty jsou dítě, žena atd.. (např. johana je dítě lucie - formule 2. - 5.).
1. $\forall X[\exists Y[dite(X,Y) \wedge zena(Y)] \wedge \exists Y[dite(X,Y) \wedge muz(Y)] \rightarrow stastny(X)]$.
2. $dite(johana, hashim)$.
3. $dite(johana, lucie)$.
4. $muz(hashim)$.
5. $zena(lucie)$.

Můžeme pozorovat, že predikátová logika má mnohem vyšší expresivitu (vyjadřovací schopnost) než logika výroková. Zásadní je především možnost modelovat vazby mezi objekty pomocí predikátů. Silným prvkem je rovněž schopnost modelovat existenci. Formule 2.-5. z příkladu mohou připomínat řádky relační databáze. Relační databáze jsou založeny na prezentaci znalostí v relacích, které modelují rovněž vztahy mezi objekty. Skutečně bychom našli jistou analogii mezi tabulkou databáze (např. tabulka dětí) a jejich atributy (kdo je dítětem koho). Zkuste si představit databázovou tabulku studentů s atributy jméno, příjmení, bydliště, ročník. Taková tabulka relační databáze se dá vyjádřit predikátem student. Databáze by pak obsahovala třeba následující řádky (a jejich ekvivalentí vyjádření v predikátové logice):
jan, novák, ostrava, 3 $\sim$ formule: student(jan, novák, ostrava, 3)
jan, veselý, ostrava, 4 $\sim$ formule: student(jan, veselý, ostrava, 4)
jiří, novák, ostrava, 1 $\sim$ formule: student(jiří, novák, ostrava, 1)
...
petr, novotný, praha, 2 $\sim$ formule: student(petr, novotný, praha, 2)

Klasické relační databáze jsou ale velmi úzkou podmnožinou způsobu reprezentace pomocí predikátové logiky. V databázích nemáme možnost používat proměnné a zejména

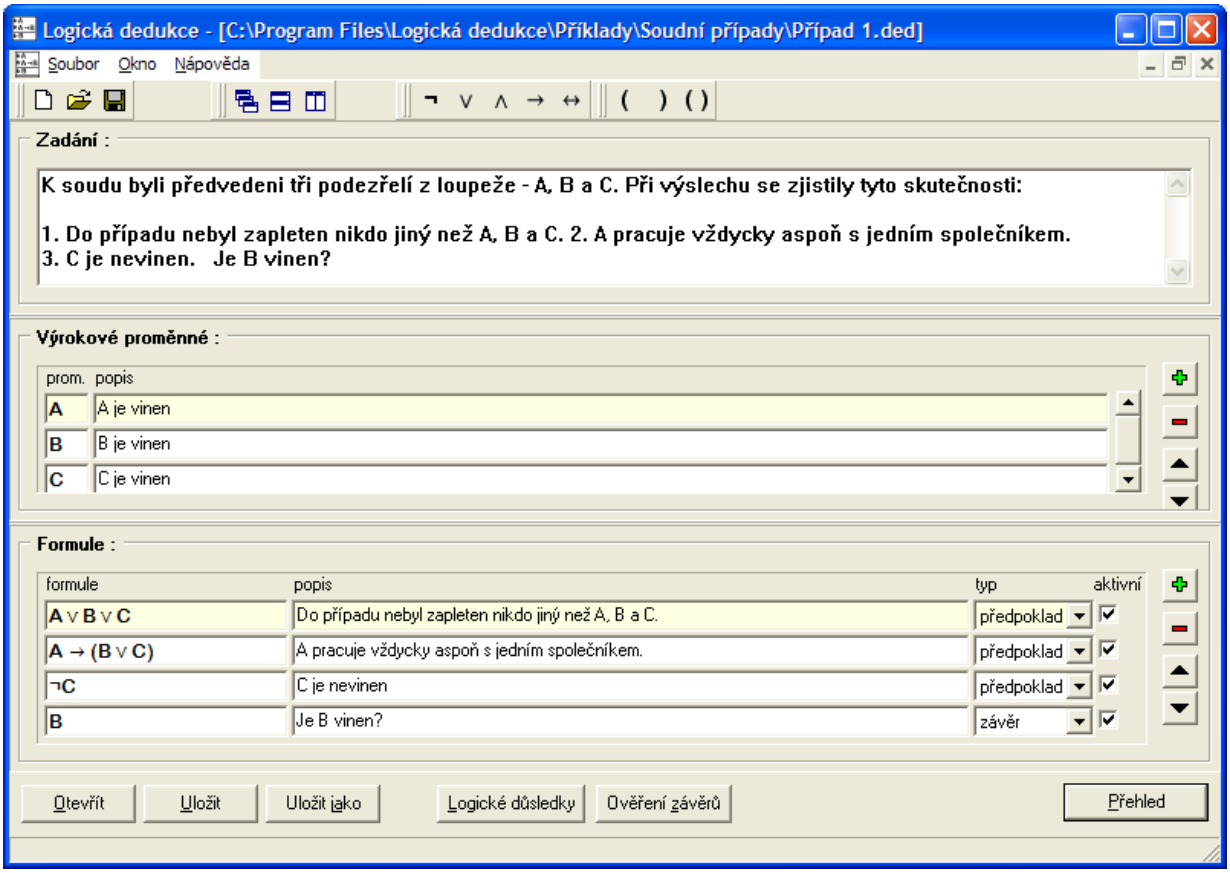
logické spojky. Ty dokáží ušetřit mnoho prostoru - co by se muselo v databázi vyjádřit explicitně (třeba tisíci relacemi), lze elegantně zapsat jedním pravidlem a aplikovat dedukci. Samozřejmě, že již dávno začaly snahy o přibližování databázové technologie a logiky a to v tzv. deduktivních databázích (např. systém Datalog). Jde o inteligentní databáze, které kromě klasického explicitního vyjmenování vztahů umožňují také používání omezených logických prostředků a dedukce.

Jak už to ale bývá téměř všude v reálném životě, za výhody se platí určitými nevýhodami. Zatímco u výrokové logiky lze vždy rozhodnout (různě efektivně) o splnitelnosti dané formule, v predikátové logice je tento problém pouze částečně rozhodnutelný. Hlavním viníkem je potenciální možnost pracovat s nekonečnými doménami objektů (např. množina přirozených čísel - lze dosazovat za proměnné jakékoliv číslo). Potenciálně existuje možnost vytvořit také o něco složitější interpretační tabulku jako u logiky výrokové, ovšem takováto tabulka může být nekonečná. V případě univerzální kvantifikace formule musí tato formule platit pro všechny možné dosaditelné konstanty. Těch ale může být ve vztahu k nekonečným doménám také nekonečně mnoho a tudíž bych dostali tabulku s nekonečným počtem řádků. Proto existují snahy predikátové logice "něco vzít", tj. odebrat jí některé vyjadřovací schopnosti při zachování některých výhod tak, aby se stala rozhodnutelná a zároveň existovaly efektivní algoritmy pro dedukci.

2 Automatizovaná dedukce

Automatizace dedukce vyžaduje najít efektivní algoritmy pro generování důsledku nebo pro prověřování konsistence množin formulí. V praktických situací jde o automatizované zjišťování, zda z logických axiómů (vybrané tautologie) a speciálních axiomů (formalizované předpoklady) vyplývá závěr. V zásadě existují **metody sémantické a formální**. Tabulky z předchozí kapitoly jsou typickou sémantickou metodou. U sémantických metod musíme provádět interpretaci formulí, což je s ohledem na již zmiňované obrovské množství ohodnocení elementárních výroků velmi neefektivní přístup. I když některé sémantické metody vylepšují tuto nevýhodu, v zásadě je sémantický přístup pro automatizaci zcela nevhodný. Druhý formální (syntaktický) přístup se snaží se zcela oprostit od interpretace (smyslu) a používat prověřená pravidla pro práci se symboly (formulemi) bez ohledu na to, co znamenají. To je v principu velice efektivní, protože časová složitost pak nezávisí na možných interpretacích, ale na velikosti předpokladů jako symbolických formulí. Tyto metody vyžadují tedy jisté "know-how", je složitější je pochopit, ale v konečném důsledku jsou zásadně lepší. Lze je rovněž naprogramovat a pak již mohou sloužit v aplikacích na "inteligentní"usuzování. Právě pro složitost těchto metod (jsou dnes zcela v režii vysokoškolské výuky) je nebudeme všechny ani naznačovat. Existuje však velmi dobře použitelná a precizně zpracovaná práce autorky Libuše Pavliskové (diplomová práce na Ostravské Univerzitě [6]). Tato práce jednak obsahuje populární výklad problematiky dedukce a zejména

je její součástí aplikace pro dedukci. Tato aplikace pracuje s výrokovou logikou (tudíž je dostatečně jednoduchá i pro středoškolskou výuku) a zároveň umožňuje zapsat libovolnou množinu předpokladů a buď generovat všechny možné důsledky nebo o konkrétním závěru zjistit, zda je to důsledek. Možná ještě významnější je pro výuku na středních školách existence velkého množství připravených příkladů s atraktivním zadáním jako jsou soudní případy podobné tomu z kapitoly 1. a další (samozřejmě i mnohem složitější). Didaktický text, popis aplikace i samotnou aplikaci pro operační systém Windows lze získat na adrese: **http://www1.osu.cz/home/habibal/dedukce/**



Obrázek 1: Grafické rozhraní aplikace pro dedukci

Formální metody dedukce lze dále v principu provádět dvěma způsoby - **přímo a nepřímo**. Zjednodušeně řečeno, při přímé metodě z předpokladů generujeme určitý důsledek pomocí odvozovacích (nebo jiných) pravidel. Při tomto způsobu tedy vždy hledáme potenciálně jiný důsledek podle toho, který závěr chceme dokázat. To v sobě samozřejmě skrývá jednu nevýhodu, jde především o možnost vygenerovat obrovské množství (potenciálně také nekonečné) různých důsledků a ten náš konkrétní se skrývá někde mezi nimi. To vede k neefektivitě a zejména se těžko hledají různé pomocné postupy, jak de-

dukci urychlit. Proto se používají spíše postupy nepřímé. Jsou podobné principu známého nepřímého důkazu. K předpokladům přidáme náš zkoumaný závěr, ovšem opačný (tedy se spojkou negace). Jelikož závěr vyplývá pokud je jeho interpretace pravda ve všech případech, kdy jsou pravdivé předpoklady, pak při opačném (negovaném) závěru taková množina nemůže být splnitelná. Při nepřímé metodě tedy s negovaným závěrem chceme dokázat nesplnitelnost. Tím se vyhneme základnímu problému přímé metody, protože náš cíl při dokazování je vždy stejný. Je jím prokázání nesplnitelnosti bez ohledu na dokazovaný závěr. To činí dedukci efektivnější. I přestože je stále obrovské množství možností, jak můžeme pomocí pravidel nakládat s předpoklady, můžeme již najít obecné postupy, jak urychlit (v určitých případech) proces hledání důkazu. Například v nepřímé rezoluční metodě (kterou si krátce vysvětlíme níže) jde o to, najít prázdnou formuli. Dá se tedy použít heuristika (metoda, která nefunguje zcela dokonale, ale v mnoha případech urychluje řešení), že je výhodnější pro následující krok hledání použít formule s co nejmenším počtem unikátních atomů (atomem se rozumí ve výrokové logice elementární výrok bez spojek). To samozřejmě není vždy pravda, ale intuitivně to je docela rozumné, pokud chceme dospět k formuli prázdné.

V tomto článku s omezeným rozsahem bychom se ještě chtěli zmínit o dvou formálních metodách. První trochu méně známá, ale i přesto velmi účinná je **metoda tablová**. Je založena na rozkladu formule ve stromu podle logických spojek a hledání větví, které obsahují navzájem negativní atomy (stejný atom s negací a bez negace). To ji činí velice účinnou, protože její časová náročnost je závislá jen na složitosti formule (počtu spojek). Bohužel v predikátové logice musí být obohacena o některé kroky, které její využití v praxi poněkud snižují. Druhou mnohem známější a široce používanou metodou je takzvaný **rezoluční princip** (rezoluce) resp. metody odvozené od něj. Myšlenku rezolučního principu formuloval v roce 1965 A. Robinson a od té doby byla velmi rozvinuta a zejména aplikována v různých systémech pro automatizované usuzování. I když existuje samozřejmě i její varianta pro logiku predikátovou, omezíme se zde pro jednoduchost pouze na logiku výrokovou. V klasickém pojetí rezoluce funguje pouze na formulích převedených do formy tzv. klauzulí (existují i zobecnění pro libovolné formule, ale zatím nejsou používány ani příliš prozkoumány). Klauzule je taková formule, kde se vyskytuje pouze binární spojka disjunkce, která spojuje jednotlivé atomy s negací nebo bez negace. Každou formuli lze pomocí logických pravidel (ekvivalencí - viz např. příklad 2) převést na ekvivalentní množinu klauzulí (může jich být i velmi mnoho). Rezoluční pravidlo pak umožňuje generovat ze dvou klauzulí obsahujících stejný atom (symbol elementárního výroku), který je v jednom případě s negací a v druhém bez negace, novou klauzuli spojenou disjunkcí obou výchozích klauzulí a zároveň vypustíme onen pár atomů, na kterých jsme prováděli rezoluci. Schématicky to lze znázornit následovně (atomy a jejich negace lze v klauzuli libovolně přesunovat bez změny interpretace dané formule).

Rezoluční pravidlo

$$\frac{C_1 \vee x \quad C_2 \vee \neg x}{C_1 \vee C_2} \tag{1}$$

C_1 a C_2 jsou zbývající části klauzulí a x je atom, na kterém rezoluci provádíme.

Máme-li pravidlo, jsme schopni z výchozích předpokladů (speciálních axiomů) a logických axiomů pomocí tohoto pravidla konstruovat sekvenci formulí, které říkáme důkaz. U každé syntaktické metody nebo formálního systému je důležité mít ověřeny dvě vlastnosti. Jde o to, aby metoda nebo systém, který obchází problematickou sémantiku tím, že pracuje pouze "slepě"se symboly bez ohledu na jejich interpretaci, byl s touto sémantikou v souladu. První vlastnost, které se říká korektnost systému, zaručuje, že používaná pravidla generují pouze logické důsledky axiomů. Kdyby tomu tak nebylo, systém by z nekorektních pravidel generoval s předpoklady zcela nesouvisející formule (nesmyslné). Druhou vlastností je tzv. úplnost systému. Ta zaručuje, abychom pouze s danou množinou pravidel a axiomů byli schopni vygenerovat veškeré možné logické důsledky. Kdyby tomu tak nebylo, měli bychom sice korektní systém, který však neumí některé důsledky generovat/ověřovat, což je jako univerzální algoritmus opět nefunkční. Pokud je systém korektní a úplný, pak se chová zcela ekvivalentně sémantice a přesto ji nijak během dedukce nemusíme uvažovat.

Nyní rezoluci aplikujeme na již sémanticky řešený soudní případ.

Příklad 5:
Nejprve je nutné formule 1. $A \vee B \vee C$, 2. $A \rightarrow (B \vee C)$, 3. $\neg C$ převést do klauzulí. Formule 1. a 3. jsou klauzulemi bez převodu. Formule 2. vyžaduje převod. Využít můžeme pravidla pro přepis implikace na disjunkci. Toto pravidlo lze schématicky zapsat jako $A \rightarrow B \Leftrightarrow \neg A \vee B$ (čtenář si může lehce ověřit pomocí tabulky, že jde opravdu o formule s totožnou interpretací). Tímto pravidlem pak formuli 2. převedeme na formuli (disjunkce je navíc komutativní a asociativní): 2. $\neg A \vee B \vee C$. Nyní již můžeme buď přímo nebo nepřímo ověřovat závěry. Nejprve se pokusíme pomocí rezolučního pravdila (rezoluce) přímo vygenerovat, že B je vinen (B). Navíc přitom použijeme pomocná pravidla (například vyskytuje-li v klauzuli atom vícenásobně, je klauzule s jedním výskytem ekvivalentní; tyto kroky označíme pomocí $\Rightarrow$). Dalším platným pravidlem je, že formule $\bot \vee F$ je ekvivalentní s F. Toto využijeme v případě, že jedna z premis obsahuje pouze jeden atom a tím pádem je po rezoluci ekvivaletní s $\bot$.

1. $A \vee B \vee C$ (axiom), 2. $\neg A \vee B \vee C$ (axiom), 3. $\neg C$ (axiom),
4. (rezoluce na A v 1. a 2.): $(B \vee C) \vee (B \vee C) \Rightarrow B \vee C$,
5. (rezoluce na C v 4. a 3. - z formule 3. nezbylo nic): B
Tím jsme provedli důkaz toho, že B je vinen (jelikož systém založený na rezoluci je korektní a úplný).

Nyní se stejný závěr pokusíme dokázat nepřímo. Přitom přidáme k předpokladům negaci závěru a budeme se snažit prokázat nesplnitelnost (to znamená vygenerovat prázdnou klauzuli, která je nesplnitelná).

1. $A \vee B \vee C$ (axiom), 2. $\neg A \vee B \vee C$ (axiom), 3. $\neg C$ (axiom), 4. $\neg B$ (negace závěru),
5. (rezoluce na B v 4. a 2. - z formule 4. nezbylo nic): $\neg A \vee C$,
6. (rezoluce na B v 4. a 1. - z formule 4. nezbylo nic): $A \vee C$,
7. (rezoluce na A v 5. a 6.): $C \vee C \Rightarrow C$,
8. (rezoluce na C v 7. a 3. - z formule 7. ani 3. nezbylo nic): $\bot$
Jelikož jsme dospěli k prázdné formuli, prokázali jsme nesplnitelnost množiny formulí 1. - 4., čímž je také prokázáno nepřímo, že B je logickým důsledkem množiny formulí 1. - 3.

Nepřímý důkaz v předchozím příkladě jsme samozřejmě mohli realizovat různými způsoby. Univerzálním přístupem je konstrukce nepřímého důkazu pomocí přímého přidáním jediného kroku, kdy provedeme rezoluci na vygenerovaný přímý důsledek a jeho negaci (pokud jde jen o atom). Na tom je vidět, že zřejmě existuje vždy mnoho způsobů, jak důkaz provést. Z hlediska časové složitosti jde principiálně o úlohu s exponenciální složitostí, o kterých jsme se již zmínili v předchozím článku v MFI. Nicméně existují přístupy, jak důkaz urychlovat a těm se říká rezoluční strategie. Některé pomáhají málo, ale jsou v principu úplné (tedy zachovávají úplnost systému) a některé jsou velmi efektivní za cenu neúplnosti (ale ve většině praktických úloh to nevadí). V některých formálních systémech je rezoluce nazývána jinak, resp. je její obecná myšlenka skryta v jiné symbolice. Například se můžete setkat s tzv. klauzulární logikou, kde se rezoluční pravidlo skrývá v tzv. pravidle řezu. Řez je výstižným pojmenováním, neboť jsme viděli, že rezoluce vlastně "vyřezává"atomy z původních formulí.

V praxi se rezoluční metoda uplatňuje především v logickém programování. **Logické programování** není tak rozšířeno jako procedurální programování (např. algoritmizace v jazyce Pascal). Při procedurálním přístupu programátor musí vymyslet způsob, jak dosáhnout řešení cíle a to pomocí řízení výpočtu (musí správně použít proměnné, přiřazování, podmínky, cykly atd.). Logické programování vychází z myšlenky automatizace dedukce. Programátor nedefinuje postup řešení, ale pouze zadává formule (pravidla a fakta), která specifikují "logiku"řešení úlohy. Na takto zapsaný program se pak může dotazovat, podobně jako při prověřování závěrů dedukce a systém sám odpoví na dotaz. Způsob řešení je univerzální a programátor se o něj nemusí starat. Jako jednoduchý příklad může sloužit výpočet faktoriálů. V procedurálním programování musí programátor vytvořit řízený výpočet, tj. musí naprogramovat cyklus nebo rekurzivní proceduru. V logickém programování stačí naprogramovat dvě pravidla (resp. jedno pravidlo a jeden fakt). Pravidlo udává hodnotu faktoriálu pro argument předchozího přirozeného čísla pomocí vazby na term s proměnnou (v logickém programování se nemá používat klasické přiřazování, měly by se používat výlučně prostředky logiky). Fakt udává hodnotu faktoriálu pro argument 0. V praxi by používání pouze logických prostředků způsobovalo neefektivní řešení, proto implementace logického programování často umožňují použití také některých omezených procedurálních prvků (např. řez). Asi nejznámějším prostředkem logického programování je jazyk PROLOG (PROgramming in LOGic). Vzhledem k omezenému rozsahu tohoto článku jej nebudeme rozebírat, ale čtenář má možnost využít elektronický učební text s mnohými příklady [2]. Pro vlastní pokusy doporučujeme získat z Internetu některou z freewarových implementací. Logické programování je výhodné především u úloh, kdy hledáme řešení v rozsáhlém stavovém prostoru a v úlohách typických v umělé inteligenci.

3 Vícehodnotová logika

Pro modelování situací v reálném světě je klasická logika poměrně chudá. Přes všechny její přednosti je zásadním problémem především dvouhodnotová logická interpretace. Již dlouhou dobu existují formalismy zavádějící například logiku trojhodnotovou, kde třetí logická hodnota kromě pravda/nepravda je "nevím". Různé pokusy o zobecnění například pomocí pravděpodobnosti byly zastíněny v šedesátých letech 20.století tzv. fuzzy matematikou. Fuzzy matematika vychází z principu fuzzy množiny, která narozdíl od klasické množiny, která buď obsahuje nebo neobsahuje prvek, může prvek obsahovat na určitém stupni příslušnosti. Prvky tedy mohou být v množině buď zcela nebo vůbec, ale také "jen trochu". Čtenář již měl možnost získat základní informace o fuzzy množinách v článku [7]. **Fuzzy logika** pak tento princip využívá tím, že rovněž interpretace formule nemusí být jen pravda nebo nepravda, ale může to být pravda v určitém stupni.

I když myšlenka fuzzy logiky je velmi prostá, lehce pochopitelná a tudíž implementovatelná do různých automatizovaných systémů, je potřeba se při jejím používání držet některých vlastností. I v běžném životě se můžete setkat s aplikacemi fuzzy logiky. Například elektrospotřebiče - pračky - mají dnes často na sobě nápis "Fuzzy-logic". Tím se může myslet například schopnost dávkovat automaticky prací prostředky nikoliv v přesně vymezených mantinelech podle váhy prádla (např. pro 1 kg prádla přidej přesně 10 g pracího prostředku), ale pouze vágními pravidly (např. je-li prádla málo, přidej pracího prostředku málo). Termínem fuzzy logika se označuje schopnost pracovat s neurčitou (vágní) informací, ale samozřejmě pojem fuzzy logika ve smyslu matematicko-informatickém je mnohem složitější. Základním problémem často bývá živelné používání množin stupňů příslušnosti bez ohledu na to, že musí existovat přímá souvislost také s používanými logickými spojkami. Proto je nezbytné, aby množina stupňů příslušnosti (pravdivosti) formule byla některou ze zavedených algeber. Množinou bývá v praktických aplikacích většinou interval [0, 1], i když teorie fuzzy množin a fuzzy logika má teoretické výsledky i pro mnohem složitější struktury. Tyto algebry mají vždy své výhody a nevýhody podle toho, jaké interpretace spojek na intervalu [0, 1] použijeme. Tyto spojky musí být ještě navzájem v souvislosti tj. použijeme-li určitou konjunkci předurčuje to již použití ostatních spojek. Nevýhodou algeber pak může být například nespojitost interpretačních funkcí spojek, neplatnost některých zásadních logických pravidel (zákonů) tak, jak je známe z klasické logiky. Pravděpodobně nejlepším kompromisem je tzv. Łukasiewiczova algebra pojmenovaná po významném polském logikovi. Tato algebra je následující struktura:

$$\mathcal{L}_{\mathrm{L}} = \langle [0,1], \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$$

kde [0, 1] je interval reálných čísel mezi 0 a 1, což jsou nejmenší a největší hodnota (nepravda, pravda). $\wedge$ a $\vee$ jsou operace infima a suprema (resp. na uvedené množině je lze ztotožnit s minimem a maximem). Definice standardních a odvozených operátorů je následující:

$$a \otimes b = 0 \vee (a + b - 1) \quad a \rightarrow b = 1 \wedge (1 - a + b) \quad a \oplus b = 1 \wedge (a + b) \quad \neg a = 1 - a$$

Na základě této algebry bychom pak mohli vytvořit příslušnou fuzzy výrokovou nebo predikátovou logiku. Zavést bychom museli kromě standardních logických spojek konjunkce

a disjunkce $\wedge$, $\vee$ (jejichž interpretační funkce by byly totožné s operacemi infima a suprema) také nové Łukasiewiczovy spojky a to Łukasiewiczova konjunkce (&) a disjunkce (∇). Jejich interpretační funkce jsou operace $\otimes$ a $\oplus$. Implikace a negace se připouští interpretovat pouze na základě operátorů této algebry.

Příklad 5:
Pomocí fuzzy logik můžeme lehce vyřešit paradoxy, se kterými si klasická logika neumí poradit. Můžete se setkat s různými formulacemi, ale v zásadě jsou všechny totožné. Známý je tzv. paradox hromady. Jde o to, že bychom v klasické dvouhodnotové logice chtěli namodelovat situaci hromady, ke které přidáváme kameny. Víme, že hromada bez kamenů je rozhodně malá. Dále je rozumný předpoklad, pokud máme malou hromadu kamenů, pak hromada s přidaným jedním kamenem bude stále malá. V klasické logice, kde jsou formule pravdivé nebo nepravdivé, by pak každá hromada byla paradoxně malá. Můžeme totiž potenciálně nekonečnou sekvencí implikací vždy dokázat, že hromada s počtem kamenů o x větším je stále malá. Ve fuzzy logice můžeme díky pravdivosti s určitým stupněm příslušnosti namodelovat tuto situaci dvěmi formulemi (použijeme predikát malahromada(t), kde t je počet kamenů v hromadě):
1. $malahromada(x) \rightarrow malahromada(x+1)$ - pravdivá ve stupni 0.999
2. $malahromada(0)$ - pravdivá ve stupni 1

Formule 1. není narozdíl od klasické logiky pravdivá zcela a díky tomu nedojde k paradoxní dedukci. Díky omezené pravdivosti bude při dedukci s každou aplikací formule 1. při navyšení počtu kamenů o 1 klesat pravdivost vyvozeného predikátu malahromada o 0.001. Chtěli bychom například ověřit stupeň pravdivosti důsledku malahromada(1) z předpokladů 1. a 2. Platí-li formule 1. na 0.999 a formule 2. na 1, pak můžeme na základě definice operátoru $\rightarrow$ interpretace (I) implikace a platného vztahu:
$I(malahromada(0)) = 1$
sestavit rovnici:
$0.999 = 1 \wedge (1 - 1 + I(malahromada(1))) \Rightarrow I(malahromada(1)) = 0.999$
Pro malahromada(2):
$0.999 = 1 \wedge (1 - 0.999 + I(malahromada(2))) = 0.001 + I(malahromada(2))$
$\Rightarrow I(malahromada(2)) = 0.998$
A tak dále pro zvětšující se počet kamenů, až pro malahromada(1000) bychom došli ke stupni pravdivosti 0.

Fuzzy logika umožňuje pracovat s neurčitou informací a je zobecněním klasické logiky. Klasická logika je vlastně speciální případ fuzzy logiky. Stejně jako mnoho jiných zobecnění, přináší i fuzzy logika řadu problémů, které klasická logika nemá. Například používání rezolučního pravidla je zde velmi omezeno, protože neexistuje univerzální postup převodu do formy klauzulí. Jde o problém aktuálně řešený například i na Ostravské Univerzitě (viz [1]). Čtenář s hlubším zájmem se může na počáteční výzkumy v této oblasti informovat v elektronicky dostupném článku (doporučujeme zejména příklady).

Zajímavé jsou také aplikace teorie fuzzy množin a fuzzy logiky, například existuje přístup s využitím tzv. lingvistických proměnných (viz [5]). Tyto proměnné mohou místo klasických číselných hodnot nabývat hodnot blízkých přirozenému jazyku jako jsou například

lingvistické výrazy typu: "velmi malý", "zhruba střední"atd. Pomocí těchto proměnných pak lze modelovat velmi srozumitelně a podobně jako člověk reálné situace. Například lze popisovat řízení auta, kde dvě z pravidel by mohla znít:
KDYŽ na semaforu svítí jen žluté světlo A ZÁROVEŇ vzdálenost od křižovatky je **malá** A ZÁROVEŇ rychlost auta je **malá** PAK sešlápni brzdu **střední** silou
KDYŽ na semaforu svítí jen žluté světlo A ZÁROVEŇ vzdálenost od křižovatky je **velmi malá** A ZÁROVEŇ rychlost auta je **střední** PAK sešlápni plyn **velkou** silou

Tato pravidla jsou pak interpretována pomocí fuzzy logiky a můžeme díky nim velmi lehce modelovat a zejména automatizovat postupy z mnoha oblastí života. Existují systémy, které se v praxi skutečně používaly a používají jako je systém LFLC (Linguistic Fuzzy Logic Controller) vyvinutý rovněž na Ostravské Univerzitě. Pomocí něj se modelovaly například technologické procesy v hutích a oproti klasických prostředkům jsou velkým vylepšením. Lze totiž na rozdíl od klasických regulačních technik, které vyžadují složitou matematiku jako jsou diferenciální rovnice a se kterými může pracovat jen velmi úzká skupina odborníků, tyto systémy svěřit i poučenému laikovi. Ten dobře zná například svůj technologický proces, který ručně obsluhoval dlouhou dobu a je schopen formulovat slovně své akční zásahy. Ty pak stačí naformulovat a odzkoušet a máme ve velmi krátkém čase funkční automatizaci daného procesu, založenou na fuzzy logice.

4 Závěr

Logika, problém dedukce a její automatizace se vyskytuje v mnoha oblastech života. Je nedílnou součástí matematiky a informatiky a proto by se výuka logiky měla odrazit nejen ve výuce matematiky na středních školách, ale právě díky problému automatizace dedukce by měla být alespoň v omezeném rozsahu i součástí informatické výuky. Výroková logika a systémy dedukce pro ni nejsou pro středoškolské studenty nedostupné, jak jsme se snažili ukázat na teorii i příkladech. S pomocí počítačových programů si student navíc může mnohem rychleji osvojit principy dokazování důsledků a to na populárních příkladech.

Logika, a to nejen vícehodnotová, ale i klasická, není v žádném případě ustrnulá disciplína. Pravý opak je pravdou a i jejich teorie se dynamicky vyvíjí právě v této době. Aplikace založené na vícehodnotové logice se dostávají již do civilního života. V současné době je velmi aktuální problém, jak lépe reprezentovat znalosti na Internetu. V současnosti zavedené možnosti reprezentace a vyhledávání informací jsou spíše syntaktického charakteru a postrádají tedy svůj smysl. Takzvaný projekt sémantického webu má za cíl dát jazyk a metody pro dedukci, které dokáží dát webovským stránkám i smysl - logiku. Jeden z nadějných pokusů je založen právě na tzv. Deskripční logice, která je v principu "odlehčenou"verzí predikátové logiky. I proto tuto aktuálnost a perspektivitu si logika a dedukce zaslouží své místo ve výuce.

Literatura

[1] HABIBALLA, H. Non-clausal resolution theorem proving for fuzzy predicate logic. výzkumná zpráva Ústavu pro výzkum a aplikace fuzzy modelování OU č. 70, 2005, dostupné na: http://ac030.osu.cz/irafm/ps/rep70.ps.gz

[2] HABIBALLA, H. Prolog. studijní text, Ostravská Univerzita, 2004, dostupné na: http://www.volny.cz/habiballa (odkaz Výuka → PROLOG)

[3] HABIBALLA, H., KMEŤ, T. Vyčíslitelnost a složitost. MFI 15 (2005 - 06), č. 2 a 3.

[4] LUKASOVÁ, A. Formální logika v umělé inteligenci. Computer Press, Brno, 2003.

[5] NOVÁK, V. Základy fuzzy modelování. BEN-technická literatura, Praha, 2000.

[6] PAVLISKOVÁ, L. Principy dedukce ve výrokové logice - výukový program. diplomová práce, Ostravská Univerzita, 2003.

[7] TALAŠOVÁ, J. Fuzzy množiny - nástroj matematického modelování vágnosti. MFI 7 (1997-98), č. 9 a 10.

[8] VOLNÁ, E. Umělá inteligence ve výuce informatiky. Sborník konference ICTE 2001, Rožnov pod Radhoštěm, 2001.

Kontaktní adresa:

Hashim Habiballa
katedra informatiky a počítačů
Přírodovědecká fakulta Ostravské Univerzity
30.dubna 22, 701 03 Ostrava 1,
tel.: +420597460213,
e-mail: habiballa@volny.cz, www: http://www.volny.cz/habiballa/