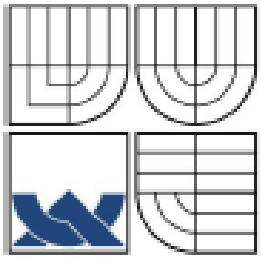
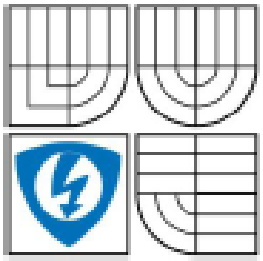


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

REPREZENTACE ZNALOSTÍ PRO EXPERTNÍ SYSTÉMY

KNOWLEDGE REPRESENTATION FOR EXPERT SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — PAVEL LEKEŠ
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR POLÁCH
SUPERVISOR

BRNO 2007

Abstract

The objective of this work is to analyze using of the first order logic for reasoning in knowledge based systems. Merit of the predicate logic is its relative simplicity in matter of algorithm development, where it is possible to decide about validity of a sentence written in first order logic language, by using formal rules without any connection to real meanings of logical variables.

In Expert systems, the first order logic is used to derivate new formulas from given axioms, where previously derived formulas are included to the axioms as well. Its process of derivation of implicit believes from explicitly given facts.

Keywords

Expert systems, Artificial intelligence, knowledge representation, predicate logic, first order logic

Anotace

Tato práce se zabývá možností použití predikátové logiky pro dedukci v systémech umělé inteligence. Výhody logiky prvního řádu jsou zejména v její relativně jednoduché algoritmizaci, kdy lze rozhodnout o platnosti či splnitelnosti věty zapsané pomocí predikátové logiky pouze pomocí formálních pravidel, tedy bez jakékoliv vazby na skutečné denotáty jednotlivých proměnných.
V Expertních systémech se predikátová logika využívá k odvození pravdivých formulí ze zadaných axiomů, přičemž se do odvozování zahrnou i nově odvozené pravdivé formule. Jedná se o proces vyhledávání implicitních znalostí z explicitně zadaných axiomů.

Klíčová slova

Expertní systémy, umělá inteligence, zpracování znalostí, predikátová logika, logika prvního řádu

Bibliografická citace

LEKEŠ, P. *Reprezentace znalostí pro expertní systémy.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 53 s. Vedoucí bakalářské práce Ing. Petr Polách.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma "*Reprezentace znalostí pro expertní systémy* " jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Děkuji vedoucímu bakalářské práce, Ing. Petru Poláchovi, za velmi užitečnou pomoc a cenné rady při zpracování a třídění poznatků.
Také bych rád poděkoval své rodině za velkou (nejen) morální podporu v průběhu celého studia.

V Brně dne : Podpis:

1. ÚVOD DO SVĚTA EXPERTNÍCH SYSTÉMŮ:

Expertní systémy (ES) zaujímají významné místo v oblasti umělé inteligence (UI). Ač se první prototypy expertních systémů objevili již v 60.tých letech 20. století a jejich komerční využití se datuje od počátku 80. let 20. století, jednotná definice pojmu expertní systém stále chybí.

Je to důsledek nejednotnosti názorů a neexistenci přesných definic pro výrazy „***inteligence***“, „***znalost***“, „***myšlení***“ a dalších pojmů, které se oboru umělé inteligence bytostně týkají. Já se však nehodlám pouštět do rozboru vášnivých filozofických debat, které zuří již od dob antického Řecka. Ve své práci, sloužící jako úvod k expertním systémům použiji jistá zjednodušení, která ovšem přispějí k lepší pochopitelnosti tématu.

Začněme tedy zjednodušením č.1 - definicí expertního systému podle Feigenbauma:

> ***Expertní systém je počítačový program, simulující rozhodovací činnost experta při řešení složitých úloh a využívající vhodně zakódovaných, explicitně vyjádřených znalostí, převzatých od experta, s cílem dosáhnout ve zvolené problémové oblasti kvality rozhodování na úrovni experta.***

K detailnější představě připojme k výše uvedené definici ještě požadavky na ně kladené, tzv. rysy expertních systémů:

- *oddělení znalostí a mechanismu jejich využívání (řídícího neboli inferenčního mechanismu).*
- *schopnost rozhodování za neurčitosti*
- *schopnost vysvětlování svých závěrů (a nejen jich).*
- *dialogový režim konzultace s uživatelem*
- *modularita uložení znalostí tak, aby bylo možné jednoduše zahrnovat přírůstky nových znalostí*
- *transparentnost znalostí, aby uložená data zůstala pro experta čitelná (pro případné úpravy), ale i pro další odborné pracovníky, kteří se jejím prostřednictvím mohou učit a doškolovat.*

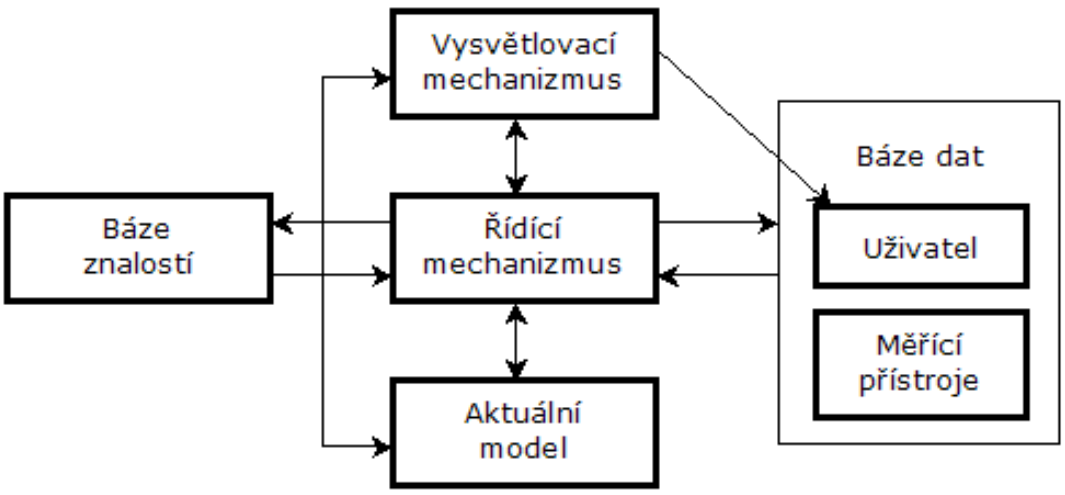
- *a další.*

Zatímco první uvedený rys bychom mohli nazvat charakteristickým, neboť je pro expertní systémy zcela stěžejní, ostatní z uvedených (i neuvedených) slouží jen jako doporučení. Zde si čtenář může povšimnout poměrně velké volnosti v posuzování co je a co není ES. Jenom publikace na toto téma by jistě zabrala několik desítek stránek.

Expertní systémy se dají rozdělit podle různých hledisek do kategorií. Nejčastěji se setkáváme s rozdělením podle **charakteru problému**, který řeší a to na systémy:

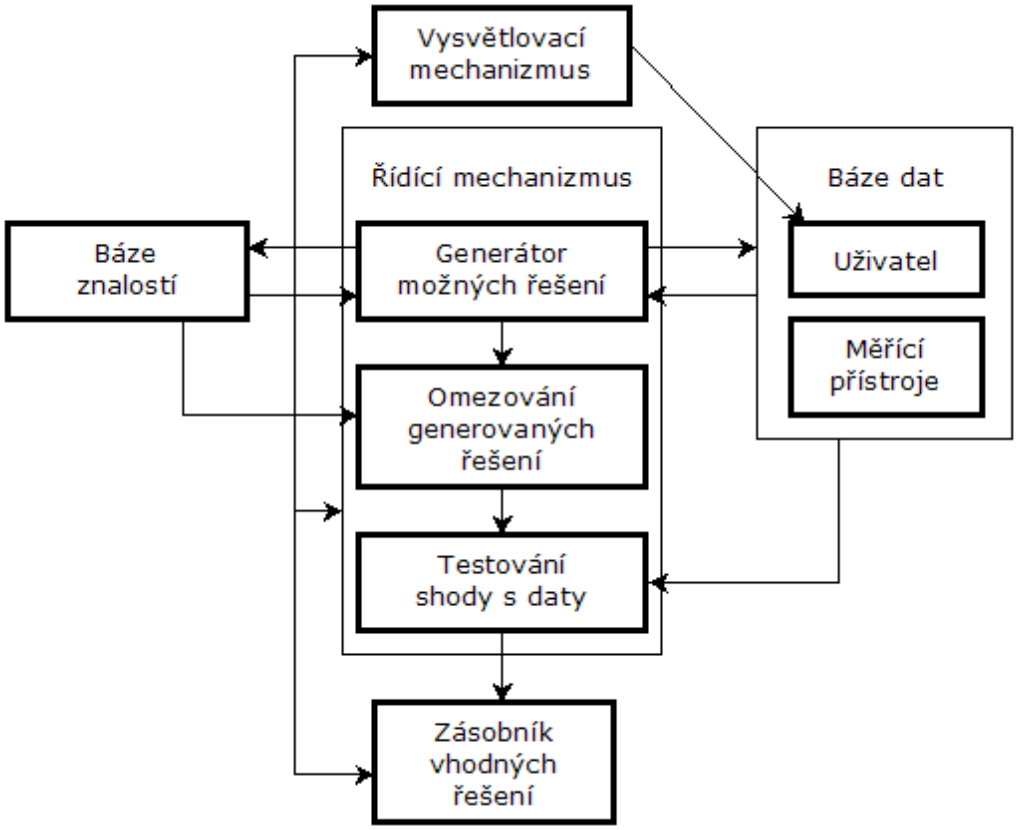
- ***diagnostické***
- ***plánovací***
- ***hybridní***

Úkolem ***diagnostických ES*** je určit, která hypotéza z předem definované konečné množiny cílových hypotéz nejlépe koresponduje s daty týkajícími se daného konkrétního případu.



Obrázek 1 Schéma diagnostického expertního systému

Plánovací ES řeší takové úkoly, kdy je znám cíl řešení a počáteční stav a je třeba s využitím dat o konkrétním řešeném případu nalézt posloupnost kroků, kterými lze dosáhnout cíle.



Obrázek 2 Schéma plánovacího expertního systému

Hybridní ES pak využívají jak diagnostického tak plánovacího přístupu.

Jak vidíme na přiložených obrázcích, můžeme o ES uvažovat jako o systému skládajícího se z modulů. Všechny moduly nemusí existovat (a většinou také neexistují) samostatně jako takové, nicméně každý ES má nejméně moduly 3 a to:

báze znalostí	kde jsou explicitně uloženy znalosti získané od experta
inferenční mechanismus	který provádí a řídí veškerou výpočtovou činnost
báze dat	vstupy od uživatele či měřících přístrojů

Báze znalostí:

Jde o množinu explicitně vyjádřených znalostí experta. Z hlediska reprezentace báze znalostí jsou na ni kladeny dva požadavky. **Požadavek modularity**, který jsme již zmiňovali, jehož hlavním úkolem je zachovat přehlednost báze znalostí pro její další případné rozšíření. **Požadavek strukturalizace**, jehož hlavním důvodem je vytvoření hierarchie pojmů, které se v bázi znalostí vyskytují. Při splnění druhého požadavku máme zajištěno snadné vyhledávání znalostí a snadný přesun od konkrétního pojmu k obecnému a naopak.

Znalosti mohou být vyjádřeny v zásadě dvojím způsobem, a to

- *deklarativně* - vyjadřují známé fakty nebo představují dotazy na konkrétní údaj a jsou proto označovány jako **poznatky**
- *procedurálně* - vyjadřují jakým způsobem se má z jednoho poznatku odvodit nějaký závěr a jsou označovány jako **pravidla**

V různých expertních systémech se tyto požadavky a způsoby různě kombinují.

Řídící (inferenční) mechanizmus

Inferenční mechanizmus je výkonná část expertního systému, která má na starosti využít dostupné znalosti, získat potřebná data a poskytnout nám odpovídající závěr. Ve většině případů jde o různé techniky prohledávání stavového prostoru, který může být vyjádřen různými způsoby - ať už pouhým výčtem pravidel či stromem, popř, grafem, objektů reprezentujících znalosti. Vlastní činnost řídícího mechanizmu se také označuje jako *inferenční (odvozovací) proces*. Proto se i řídící mechanizmus někdy označuje jako *inferenční mechanizmus.*

Kromě těchto základních algoritmů se ale používají ještě různé doprovodné techniky, které přímo nesouvisejí s vlastním prohledáváním stavového prostoru, ale mohou

jistým způsobem pozitivně ovlivňovat činnost řídícího mechanizmu. Jde například o *Agendu, Démony, Tabule, Taxonomie a další*

K odvozování nových údajů může řídící mechanizmus využít následující metody (někdy bývají označovány jako **inferenční metody**):

- **Dedukce** - Odvozování závěrů, které přímo plynou z předpokladů.
- **Indukce** - Odvozování obecnějších poznatků ze specifičtějších poznatků.
- **Generování a testování** - Získávání poznatků metodou pokus-omyl.
- **Abdukce** - Zpětné usuzování ze známého pravdivého závěru směrem k jeho předpokladům.
- **Absence** - Pokud chybí patřičná konkrétní znalost, implicitně se předpokládá nějaká obecná znalost.
- **Analogie** - Získávání závěrů na základě podobnosti situace vzhledem k jiné známé situaci.

Existuje samozřejmě bezpočet dalších používaných technik, ovšem další text bude zaměřen zejména na **dedukci**. Nicméně abychom se mohli zabývat – byť tou nejzákladnější inferenční metodou – trochu blíže, je třeba se podívat na způsob uložení faktů v **bázi znalostí.**

K tomu, abychom mohli blíže prozkoumat **bázi znalostí** ovšem potřebujeme vědět, jak taková znalost „vypadá“. Jednoduše řečeno, je třeba probrat se základy nějakého logického jazyka, který nám umožní vyjádřit znalosti v počítači pochopitelné formě.

Kapitolu 2 tvoří základy jednoduchého (alespoň podle všech UI odborníků) jazyku **First Order Logic**. Po jejím absolvování byste měli vědět, co říká například věta: $\forall x Na[(x,VUT) \supset Chytrý(x)]$

2. JAZYK FIRST ORDER LOGIC

Dříve, než může systém usilující o určitou úroveň inteligentního chování začít uvažovat, učit se, plánovat nebo vysvětlovat své chování, musí být schopen vyjádřit příslušné myšlenky. Je tedy třeba použít nějaký programovací jazyk, který mu to umožní. Jednou z možností je použití predikátové logiky - **First Order Logic** (dále v textu jej budu označovat zkratkou FOL).

V rámci programovacího jazyka používaného v UI (a vlastně i jakéhokoliv jiného) je kladen důraz ja syntaxi, sémantiku a pragmatiku.

Syntaxe:

Je třeba specifikovat jaké symboly a jak seřazené budeme považovat za platnou formulaci. Tak například věta „Včera bylo venku velmi větrno“ je (v češtině) správně zapsaná oznamovací věta, ovšem větu „Velmi včera větrno bylo venku“ už za správně složenou jistě pokládat nemůžeme. Samozřejmě, úroveň naší inteligence nám umožňuje pochopit význam předchozí věty, ovšem každý kdo se někdy setkal s programováním, může ze svých (většinou hořkých zkušeností) potvrdit, že počítače nám takto ležérně zapsané příkazy rozhodně neodpustí. Co se reprezentace znalostí týče, je třeba naprosto jasně určit, co jsou přikazy jazyka, neboť právě ty vyjadřují logické výrazy a vztahy.

Sémantika:

Musíme specifikovat, co všechny platné výrazy znamenají. U příkazů si musíme být jistí, jaký názor na svět vyjadřují. Existují gramaticky správně poskládané věty, které žádný význam nemají. „Těžce stravitelná lžíce sešitu“. Je tedy třeba definovat, které správně poskládané výrazy význam nesou a které ne.

Pragmatika:

Je potřeba specifikovat, jak budou správně vytvořená a smysluplná vyjádření použita. Jinak rečeno je pragmatika o potírání dvou a vícesmyslných výrazů. Pro příklad jistě nemusíme chodit daleko. Například vyjádření „Teď jsi to opravdu

vylepšil“ může na jedné straně vyjadřovat pochvalu, na straně druhé je to výraz přímo překypující ironií. Jelikož v češtině o dvojsmyslné výrazy rozhodně není nouze, další vysvětlování je myslím zbytečné.

2.1 SYNTAXE JAZYKA FOL

V jazyce FOL existují 2 druhy symbolů. ***Logické*** (logical) a ***obecné*** (nonlogical).

Logické **(**logical**)** výrazy jsou ty, které mají pevně daný význam nebo použití. Rozlišujeme 3 druhy logických výrazů:

1) **interpunkce:** „**(**“, „**)**“ a „**.**“
2) **operátory:** „¬“ pro *logickou negaci*, „∧“ pro *logický součin* („and“), „∨“ pro *logický součet* („or“), „∃“ značí *existenci*, „∀“ znamená „*každý prvek*“ a „=“ znamená *logickou rovnost*. Znaky „∃“ a „∀“ nazýváme kvantifikátory.
3) **proměnné:** nekonečný počet symbolů, obecně označené **x, y** a **z,**

Obecné (nonlogical) symboly jsou takové, jejichž význam nebo použití zavisí na konkrétní aplikaci.

1) **funkční symboly** (function symbols):

Nekonečná zásoba symbolů, které budeme psát s počátečním malým písmenem, kde každé každé další slovo bude odděleno velkým počátečním písmenem (například nejlepšíKamarád, nejhezčíKočkaMéhoBratra atd.), které budu obecně označovat jako a, b, c, f, g a h (s případnou indexací).

2) **predikátní symboly** (predicate symbols)

Nekonečná zásoba symbolů, které budu psát se všemi počátečními velkými písmeny. Například StaršíNež nebo MenšíNež a které budeme formálně označovat symboly P, Q a R s případným použitím indexace.

Jedním z charakteristických prvků **obecných** symbolů je tzv. ***arita***.

Arita je nezáporné celé číslo, označující počet parametrů, které symbol přebírá.

Například *StaršíNež* je predikátní symbol s aritou 2 (Přijímá 2 parametry – obecně X, Y – které seřadí podle patřičné relace)

Jazyk FOL zná 2 typy syntakticky platných vyjádření:

Objekty (terms) a **vztahy** (formulas). Objekty se vztahují k entitám ve světě a vztahy se používají pro vyjádření relací mezi členy.

Množina objektů musí splňovat přinejmenším tyto podmínky:

- každá proměnná je objekt.
- pokud $t_1,\ldots,t_n$ jsou podmínky a f je funkční symbol arity n, pak $f(t_1,...,t_n)$ je také objekt

Množina vztahů musi splňovat přinejmenším tyto podmínky:

- pokud $t_1,\ldots,t_n$ jsou podmínky a P je predikátní symbol arity n, pak $P(t_1,...,t_n)$ je logický vztah
- pokud t_1 a t_2 jsou objekty (terms) pak $t_1 = t_2$ je vztah
- pokud α a β jsou vyjádřením vztahů a x je proměnná, pak $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $\forall x.\alpha$ a $\exists x.\alpha$ jsou vzorce.

Vztahy typu 1 a 2 (neobsahující žádné jednodušší vzorce) se nazývají atomy.

Zde je na místě uvést, že v dalším textu budou použity jisté zkratky, jako je např. vynechávání prázdných závorek, teček, použití hranatých a složených závorek apod. Občas také vynechám závorky na základě předpokladu, že $\wedge$ má přednost před $\vee$ (tak jako má ***** přednost před +) nebo při zápisu predikátních symbolů arity 0 (kde neexistuje argument, který by byl uzavřen v závorkách). Takto zkrácené zápisy neztratí nic ze své platnosti, naopak budou daleko lépe čitelné a tím i pochopitelné.

Propoziční podmnožinou jazyka FOL budeme rozumět jazyku bez objektů a kvantifikátorů, kde jsou použity pouze propozicionální (predikátní) symboly.

Například $(P \wedge \neg(Q \vee R))$

Kde P, Q a R jsou predikátní symboly by tvořil platný vzorec.

Také použijeme následující substituce:

- $(\alpha \supset \beta)$ pro $(\neg \alpha \vee \beta)$;
- $(\alpha \equiv \beta)$ pro $((\alpha \supset \beta) \wedge (\beta \supset \alpha))$;

Dále je třeba zmínit se o poli působnosti kvantifikátorů. Říkame, že proměnná je vázaná ve vzorci, pokud se vyskytuje v poli působnosti kvantifikátoru. Jinak na ni pohlížíme jako na volnou proměnnou. TZN, že proměnná x je vázaná, pokud se objeví ve spojení $\forall x.\alpha$ nebo $\exists x.\alpha$. Například ve vzorci:

$\forall y.P(x) \wedge \exists x[P(y) \vee Q(x)]$

jsou oba dva výskyty y vázané. Co se x týče, první výskyt je volný a další 2 jsou vázané.

Věta ve FOL je jakýkoliv vztah, který neobsahuje žádné volné proměnné.

Věty jazyka FOL jsou tím, co používáme k vyjádření znalosti. Zbývající vyjádření slouží pouze k podpoře syntaxe.

2.2 SÉMANTIKA:

U sémantiky nastává jistý problém. Nemůžeme totiž jednou provždy určit význam určité věty ve FOL a to z jednoho prostého důvodu. Obecné symboly jsou aplikačně zaměřené a těžko se proto můžeme dohodnout na globálním významu těchto symbolů. Tak například se nemůžeme shodnout na tom, co přesně věta „Šťastný(karel)“ říká o světě. A to ani tehdy, pokud se shodneme na tom, co znamená vlastnost „Šťastný“. Jediné, na čem se můžeme shodnout, je to, že jedinec

jménem „Karel“ (ať už je to kdokoliv) má vlastnost pojmenovanou „Šťastný“ ať už je to cokoliv. Jinak řečeno, můžeme se shodnout jen na tom, jak se odvodí význam věty z použitých obecných symbolů.

Význam věty je funkce interpretace predikátních a funkčních symbolů.

Vzhledem k výše uvedenému určení, se musíme na svět dívat zjednodušeně:

1) Ve světě jsou objekty

2) Pro každý predikát arity 1, některé objekty P uspokojí a jiné ne. Interpretace P stanoví patřičnou otázku a rozhoduje, zda objekty mají nebo nemají vlastnost, na kterou se otázka ptá.

3) Ostatní aspekty světa nehrají žádnou roli.

Pojetí FOL je takové, že výše uvedené předpoklady je všechno co je potřeba určit, co se významu obecných symbolů a tudíž významu všech vět týče. Významem predikátního symbolu „DemokratickáZemě“ budou ty země, které my považujeme za demokratické. Kdokoliv jiný může s námi vybraným seznamem zemí nesouhlasit, ovšem to už je jen a jen věcí interpretace a osobního pojetí každého člověka.

2.3 INTERPRETACE:

Významy jsou určeny **specifickou interpretací** a je třeba je přesně popsat. Interpretace $\mathfrak{I}$ tvoří v jazyku FOL pár <*D*,*I*> kde *D* je neprázdná množina objektů, kterou nazýváme oblast nebo **doména** (domain) **interpretace** a *I* je množina objektů k nim přiřazených, kterou nazýváme **interpretační zobrazení** (interpretation mapping) od obecných symbolů k funkcím a vztahům nad D jak bude popsáno později. Predikátním symbolům přiřadí interpretační zobrazení I význam následujícím způsobem:

Každý predikátní symbol P arity n, I[P] je n-násobná relace nad D, což se zapisuje

$$I[P] \subseteq D_1 \times \ldots \times D_n.$$

Jako příklad si uveďme predikátní symbol arity 1, **Pes**. Zde bude **I[Pes]** nějaká podmnožina D, pravděpodobně podmnožina psů, kteří vyhovují dané interpretaci D. Podobně I[StaršíNež] bude podmnožina D × D, pravděpodobně množina dvojic objektů, kde je první objekt starší než objekt druhý.

Funkčním symbolům přiřadí interpretační zobrazení I význam následujícím způsobem:

Každý funkční symbol f arity n, I[f] je n-násobná funkce nad D, což se zapisuje

$$I[f] \subseteq D_1 \times \ldots \times D_n \rightarrow D$$

Takže například I[NejlepšíPřítel] bude funkce [D → D], pravděpodobně funkce, která přiřazuje osobu k jejímu nejlepšímu příteli.

K vypořádání se s členy obsahujícími proměnné, ještě potřebujeme vyřeši přiřazování proměnných nad D, což je přidělení proměnných k elementům D. Pokud je μ přiřazení proměnných a **x** je proměnná, pak je μ**[x]** elementem domény D. Formálně zapsáno, za dané interpretace **I** a přiřazení proměnné μ, označení termu t, zapsáno $\|\mathbf{t}\|_{\mathfrak{I},\mu}$ je definováno těmito pravidly:

1) pokud je x proměnná, pak $\|x\|_{\mathfrak{I},\mu} = \mu[x]$;

2) pokud $t_1,\ldots,t_n$ jsou podmínky a f je funkční symbol arity n, pak

$$\|f(t_1,\ldots,t_n)\|_{\mathfrak{I},\mu} = F(d_1,\ldots,d_n)$$

kde $F = I[f]$ a $d_i = \| t_i \|_{\mathfrak{I},\mu}$

Všimněme si, že podle těchto rekurzivních pravidel je $\| t_i \|_{\mathfrak{I},\mu}$ vždycky elementem D.

Satisfakce a modely:

Máme-li definovanou interpretaci $\mathfrak{I}$<**D,I**> a substituci $\|\cdot\|_{\mathfrak{I},\mu}$ můžeme nyní určit, které věty jazyka FOL jsou **true** a které **false**. Například, „Pes(nejlepšíPřítel(honzaBrzlík)“ bude true v $\mathfrak{I}$ tehdy a jenom tehdy, pokud se bude v $\mathfrak{I}$ vyskytovat podmnožina psů a objekt, který označuje funkce „nejlepšíPřítel(honzaBrzlík)“.

Formálně řečeno, za dané interpretace $\mathfrak{I}$ a přiřazení proměnných μ, říkáme, že vztah ***α je uspokojen*** v $\mathfrak{I}$, zapsáno $\mathfrak{I} \models \alpha$, pokud splňuje následující podmínky:

Předpokládejme, že $t_1,\ldots,t_n$ jsou terms, P je predikátní symbol arity n, α a β jsou vzorce a x je proměnná.

1) $\mathfrak{I},\mu \models P(t_1,\ldots,t_n)$ iff $< d_1,\ldots,d_n > \in P$, kde $P = I[P]$ a $d_i = \| t_i \|_{\mathfrak{I},\mu}$
2) $\mathfrak{I},\mu \models t_1 = t_2$ pokud $\| t_1 \|_{\mathfrak{I},\mu}$ a $\| t_2 \|_{\mathfrak{I},\mu}$ jsou stejným elementem D
3) $\mathfrak{I},\mu \models \neg\alpha$ pokud neplatí, že $\mathfrak{I},\mu \models \alpha$
4) $\mathfrak{I},\mu \models (\alpha \wedge \beta)$ pokud $\mathfrak{I},\mu \models \alpha$ a $\mathfrak{I},\mu \models \beta$
5) $\mathfrak{I},\mu \models (\alpha \vee \beta)$ pokud $\mathfrak{I},\mu \models \alpha$ nebo $\mathfrak{I},\mu \models \beta$ nebo oboje
6) $\mathfrak{I},\mu \models \exists x.\alpha$ pokud $\mathfrak{I}, \mu' \models \alpha$ pokud se μ' odlišuje od μ ve většině x
7) $\mathfrak{I},\mu \models \forall x.\alpha$ pokud $\mathfrak{I}, \mu' \models \alpha$ pokud je μ' shodná s μ ve všech x

Pokud je α věta, je zřetelně vidět, že satisfakce nezávisí na daném přiřazení proměnných (věty jazyka FOL nemají volné proměnné). V tomto případě píšeme $\mathfrak{I}\models\alpha$ a říkáme, že α je true v interpretaci $\mathfrak{I}$ jinak je false.

Budeme také používat zápis $\mathfrak{I} \models S$, kde S je množina vět, abychom vyjádřili že S jsou pravdivé v $\mathfrak{I}$. Říkáme, že $\mathfrak{I}$ je logický model S.

2.4 LOGICKÝ DŮSLEDEK:

Všimněme si, že ačkoliv sémantické pravidla interpretace závisí na interpretaci obecných symbolů, interpretace logických operátorů mezi větami FOL jsou pevně dány.

Mějme například věty α a β a mějme větu γ, pro kterou platí: $\gamma = \neg (\beta \wedge \neg \alpha)$ Předpokládejme, že $\Im$ je nějaká interpretace, kde α = true. Nyní, za použití předchozích pravidel vidíme, že γ musí být taktéž true. (Hodnota logického součinu $\beta \wedge \neg \alpha$ je false a γ se tedy rovná $\neg$false, což je true). Tento důsledek je nezávislý na jakékoliv interpretaci obecného symbolu. Pravdivost γ vyplývá z pravdivosti α. Říkáme, že γ je logickým důsledkem α.

Mějme soubor vět S a nějakou větu α. Říkáme, že α je logickým důsledkem S, což zapisujeme jako S $\models \alpha$ pokud pro každou interpretaci $\Im$ platí: pokud $\Im \models$ S, pak $\Im \models \alpha$. Jinými slovy, pokud každý model S vyhovuje α.

3. BÁZE ZNALOSTÍ

Nyní už se dostáváme k uvažování, vyvozování, usuzování, dedukci. Teď už si zcela jistě dovedeme představit systém, který by z faktu, že Bohouš je pes mohl odvodit, že je to savec, masožravec, šelma …. Prostě jakýkoliv jiný fakt, který o psech můžeme říct. (Záměrně jsem ve svém příkladu zůstal u zcela jednoznačných faktů. Mohl jsem jako příklad uvést i to, že má Bohouš 4 nohy, nicméně jistě si dovedeme představit svět, kde Bohouš – z jakéhokoliv důvodu – 4 nohy nemá).

Řekněme tedy, že to, co doopravdy chceme, je systém, který by dokázal z faktu „Pes(Bohouš)“ vyvodit závěr „Savec(Bohouš)“. To už se ovšem pohybujeme mimo oblast logických důsledků! Existují takové interpretace, kde bude „Pes(Bohouš)“ true a „Savec(Bohouš)“ false. Abychom dosáhli požadovaného propojení, je třeba do množiny vět S zahrnout výrok, které oba symboly dává do vzájemné relace. V našem případě (resp. případě FOL) je to věta $\forall$**x.Pes(x)** $\supset$ **Savec(x),** která říká, že každé x, které vyhovuje predikátnímu symbolu Pes vyhovuje také predikátnímu symbolu Savec. Jednoduše řečeno, každý pes je savec.

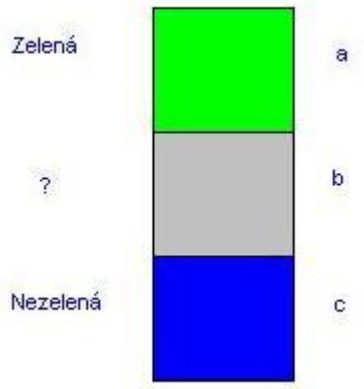
Tím, že jsme do množiny definic našeho světa zařadili větu $\forall$x.Pes(x) $\supset$ Savec(x), jsme zamítli všechny interpretace, kde množina psů nebyla podmnožinou savců. Pokud budeme do souboru S přidávat víc a víc takovýchto vět, vyřadíme tím víc a víc nežádoucích / nezamýšlených důsledků a náš systém se začne přibližovat k tomu, co chceme dosáhnout a bude ukazovat „pravdu takovou, jakou ji chceme“.

> ***Uvažování, založené na logickém důsledku zajišťuje spolehlivé a zaručené výsledky. Při použití bohaté zásoby vět jako předpokladů, zahrnujících nejen fakta, o jednotlivých objektech, ale i vztahy mezi nimi, bude množina logických důsledků o hodně bohatší, blížící se k zobrazení světa tak, aby se blížil naší představě. Výpočet důsledků se pak bude jevit jako uvažování někoho, kdo rozumí významu použitých výrazů.***

3.1 EXPLICITNÍ A IMPLICITNÍ PŘESVĚDČENÍ

Soubor vět, sloužících jako základy pro vyvozování důsledků jsou tím, co nazýváme bází znalostí. V našem případě je to určitá množina vět jazyka First Order Logic. Úkolem expertního systému je vyvodit z daných premis logické důsledky. Na bázi znalostí se tedy můžeme dívat jako na soubor znalostí, které jsou systému explicitně předány, zatímco logické důsledky můžeme brát jako implicitní znalosti. To, že systému předáme „bohatou" zásobu explicitních znalostí, ještě neznamená, že jsme udělali všechnu práci a na žádné uvažování už nezbylo místo. Krásy a složitosti uvažování uvidíme v následujícím příkladu.

3.1.1 Příklad přechodu od explicitních znalostí k implicitním:



Obrázek 3

Uvažujme situaci znázorněnou na obrázku č.3. Předpokládejme, že máme 3 barevné kostky poskládané jako věž, jednu na druhé. O nejvyšší kostce víme, že je zelená, o spodní kostce víme, že není zelená a o prostřední nemáme vůbec žádné informace. Otázka zní, jestli se v naší věžičce vyskytuje zelená kostka ležící přímo na nezelené.

Zde už je potřeba trochu přemýšlení. Vyjádříme si náš příklad v jazyce FOL.

Jednotlivé kostky si označíme jako **a, b, c**. Predikátní symbol **Z** bude sloužit jako označení barev. **Z** bude značit zelenou barvu a ¬**Z** barvu jinou. Predikátní symbol **N** bude určovat kostky ležící přímo na sobě. Fakta, která máme v S, jsou následující:

{N(a,b), N(b,c,), Z(a), ¬Z(c)} A to je vše co potřebujeme.

Ptáme se na to, jestli z výše uvedených čtyř faktů vyplývá, že v naší věži leží zelená kostička na kostičce nezelené, neboli jestli S |= α, kde α je

$\exists x \exists y.Z(x) \wedge \neg Z(y) \wedge N(x,y)$.

Potřebujeme dokázat, že každá interpretace, která uspokojuje S, vyhovuje také α. Nechť je $\Im$ jakákoliv interpretace kde $\Im$ |= S. Existují 2 možnosti, které musíme prozkoumat:

1) Předpokládejme, že $\Im$ |= Z(b) (aneb prostřední kostka je zelená). Vzhledem k tomu, že vyjádření S obsahující ¬Z(c) a N(b,c) je následující

$$\Im \models Z(b) \wedge \neg Z(c) \wedge N(b,c)$$

můžeme vidět, že se rovná našemu obecnému vyjádření α:

$$\Im \models \exists x \exists y.Z(x) \wedge \neg Z(y) \wedge N(x, y)$$

2) Musíme však ještě prozkoumat opačnou situaci, tzn. $\Im$ |= ¬Z(b) (aneb prostřední kostka má jinou než zelenou barvu). A protože vyjádření S zahrnující Z(a) a N(a,b) vypadá následovně

$$\Im \models Z(a) \wedge \neg Z(b) \wedge N(a,b)$$

opět můžeme prohlásit, že toto vyjádření uspokojuje α

$$\Im \models \exists x \exists y.Z(x) \wedge \neg Z(y) \wedge N(x, y)$$

V každém případě, je to případ kde $\Im$ |= S, to znamená, že **α je logickým důsledkem S.** Mohli jsme vidět, že i velmi jednoduché příklady, jako byl tento, mohou vyžadovat propracovanou formu uvažování.

3.2 VYJADŘOVÁNÍ ZNALOSTÍ:

Nyní máme připravenou situaci pro detailnější pohled na vytváření Báze znalostí. Báze znalostí bude obsahovat množinu vět, které budou vyjadřovat stav našeho světa. Navrhování báze znalostí má na starosti **znalostní inženýrství** (knowledge engeneering). Při návrhu báze znalostí se je třeba se zamyslet nad tím, za jakým účelem ji vytváříme – co vlastně chceme, aby zahrnovala. Vymezit objekty, které jsou pro nás důležité, vlastnosti, které dané objekty budou mít, vztahy mezi nimi atd. Pokud někdy bude zmíněna ***ontologie,*** má se na mysli právě toto. Pojďme se tedy poněkud detailněji podívat, jak se tvoří báze znalostí.

První, co musíme udělat je zvolit si nějaký svět, pro který vytvoříme množinu vět S, které jej budou definovat. Nejlépe svět, který si dokážeme velmi dobře představit. Pojďme si vytvořit svět malého městečka, které je zavaleno množstvím skandálů, plné překroucených a různě propletených vztahů. Městečko, ze kterého by měl radost každý režisér druhořadých televizních seriálů.

3.3 SLOVNÍK:

Není od věci začít výběrem slov a slovních spojení (predikátních a funkčních symbolů v rámci jazyka FOL). Nejdříve začneme se jmény osob/herců v našem světě. V jazyce FOL je budou zastupovat konstanty jako například KarelVondrák, LiborKren, LenkaKonecna atd. Mimo lidí by v našem modelu městečka mohla být také zvířata, roboti, duchové apod.

Další třídou by mohli být právní subjekty, jako jsou pojišťovny (nejlepšíZP), úřady (městskýSoud), restaurace (hospodaNaMytince) atd. K nim bychom přiřadili další důležitá místa, jako „LeehoDům“, „opuštěnýDůl“, „starýTábor“ atd.

Dále nesmíme opomenout na věci, které hrají v našem světě důležitou roli, jako například „břitva1“, „prsten5“ atd (je běžným zvykem používat čísla k rozlišení předmětů, které nemají jednoznačný název. Břitev a prstenů bude v našem městečku jistě spousta)

Po výčtu objektů hrajících více či méně důležitou roli v našem světě, je důležité určit výčet základních typů, které námi určené objekty popisují. To se provádí unárními predikátnímy symboly (predikáty s aritou 1). Pohledem na výčet našich objektů bychom se jistě neobešli bez predikátů Muž(x), Žena(x), Místo(x), Společnost(x), Šperk(x) a Nůž(x). A to má množina našich objektů jenom 12 prvků!

Další množinou unárních predikátních symbolů, bez kterých se neobejdeme je množina prvků popisujících vlastnosti jednotlivých objektů. Pomocí nich řekneme, zdali je daný objekt Bohatý(x), Milý(x), Atraktivní(x), Zpocený(x), atd.

Další predikáty, které použijeme, budou vyjadřovat vztahy mezi entitami našeho světa. Zde se žádná podmínka na aritu neklade. Vztahy bývají popsány predikátními symboly arity n. ŽijeV, JeDcerou, ŽenatýS, Pomlouvá, MělaRománekS, SpiknulSeS…

Nakonec je třeba zachytit funkce, hrající důležitou roli v námi zvolené oblasti. Například jeOtec, nejlepšíPřítel, šéf atd.

3.4 ZÁKLADNÍ FAKTA:

Pokud máme hotovou ontologii, můžeme naši bázi znalostí začít plnit fakty o světě, který jsme si vytvořili. Fakta bývají reprezentována atomy a jejich negacemi. Malé připomenutí: **atomy nazýváme vzorce neobsahující žádné jednodušší výrazy**. Můžeme začít tím, že každému objektu přiřadíme jeho typ: Muž(karel), Žena(jana), Společnost (nejlepšíZP) atd. Jakmile přiřadíme všem objektům jejich typ, můžeme se vrhnout na přiřazení vlastností jednotlivým entitám. Tato přiřazení jsou stěžejní, protože nás pak bude nejvíc zajímat to, jaké vlastnosti a vztahy vyplývají z faktů obsažených v bázi znalostí. V naší ukázce bychom mohli použít následující přiřazení. *Bohatý(karel)*, *¬ŠťastněVdaná(jana)*, *PracujePro(petr, nejlepšíZP)*, *Krvavá(břitva)* atd. Další fakty bychom mohli popsat pomocí logické rovnosti a unární funkce:

nejlepšíPřítel(karel) = *petr* by například zachycoval fakt, že Petr je Karlův nejlepší přítel. Dále bychom mohli znak = použít v případech, kdy chceme zachytit, že některý z objektů má více než jedno jméno, *NZP* = *nejlepšíZP* atd.

3.5 KOMPLEXNÍ FAKTA

Spousta faktů, které bychom o našem světě rádi vyjádřili, jsou natolik spletité, že je nebudeme schopni popsat pomocí nejjednodušších vět – atomů. Budeme si tedy muset pomoci použitím kvantifikátorů a operátorů.

Řekněme, že v našem světě všichni bohatí muži milují Janu. Abychom to popsali, musíme použít univerzální kvantifikaci, která ze všech objektů vybere bohaté muže:

$$\forall y[\text{Bohatý}(y) \wedge \text{Muž}(y) \supset \text{Miluje}(y, \text{jana})]$$

Bohaté muže jsme zde popsali konjunkcí predikátů Bohatý a Muž. Podobně bychom mohli zapsat, že všechny ženy kromě Jany milují Karla:

$$\forall y[\text{Žena}(y) \wedge y \neq \text{Jana} \supset \text{Miluje}(y, \text{karel})]$$

Pomocí univerzální kvantifikace můžeme zapsat i velmi obecná fakta

$$\forall x \forall y[\text{Miluje}(x,y) \supset \neg\text{Pomlouvá}(x,y)]$$

čímž bychom vyjádřili fakt, že ten, kdo někoho miluje, svého miláčka nepomlouvá. Ve všech předchozích příkladech byla názorně vidět užitečnost kvantifikátorů. Ekvivalentní zápis bychom vytvořili i bez nich, ovšem museli bychom vyjmenovávat všechny zahrnuté objekty. V komplexnějším světě je takový přístup poměrně těžko představitelný. Ba co víc, každá další změna v objektech, by znamenala přepsání všech vět, které by měly daný objekt zahrnovat, zatímco univerzální kvantifikátor zahrnuje všechny objekty explicitně, „sám od sebe“.

Další typ faktů, vyžadující složenou větu je ten, který vyjadřuje ***nekompletní znalost.*** Představte si, všudypřítomné klepny zjistily, že se nám Jana zamilovala. Identita její nové lásky je však nejasná. Buďto se jedná o Michala nebo Petra. Zápis ve FOL?

$$\text{Miluje}(\text{jana}, \text{michal}) \vee \text{Miluje}(\text{jana}, \text{petr})$$

Podobně, kdybychom věděli, že nějaká žena pomlouvá Karla, použili bychom k vyjádření této neurčité / nekompletní znalosti existenčního kvantifikátoru

$$\exists x[\text{Žena}(x) \wedge \text{Pomlouvá}(x,\text{karel})]$$

Za předpokladu platnosti předchozích vět je zcela jasné, o kterou „dobračku“ jde. V tomto případě je už použití existenčního kvantifikátorů nezbytné. Pomocí klasických operátorů totiž nejsme s to vyjádřit neurčitost.

Další typ vět používaný v bázi znalostí bychom mohli nazvat „uzavřená věta“. Jedná se ve své podstatě o výčet, který omezuje oblast, kterou máme na mysli. Mohli bychom například udělat výčet všech právníků v našem světě.

$$\forall x[\text{Právník}(x) \supset x = \text{jana} \vee x = \text{david} \vee x = \ldots\ldots]$$

nebo bychom mohli definovat všechny sezdané páry

$$\forall x \forall y[\text{ŽenatýS}(x,y) \supset (x = \text{eliška} \wedge y = \text{vikor}) \vee \ldots.]$$

Poté by vyplývalo, že každý partnerský pár, který není **výslovně** uveden v předchozí větě, žije tzv. na hromádce (já vím, co je komu do toho, ale držme se scénáře prosím) Obecně vzato bychom takto mohli vyjmenovat všechny objekty v našem světě:

$\forall x[x=\text{jana} \vee x = \text{břitva} \vee x = \text{opuštěnýDůl} \vee \ldots.]$

což by zaručovalo, že se během upravování databáze nebo procesu uvažování neobjeví na scéně nějaký neznámý či nezamýšlený objekt.

Jako poslední z faktů si zde uvedeme formální rozlišení objektů např. jana $\neq$ karel, čímž předejdeme náhodnému označení 2 osob (věcí, míst) jako jedné entity.

3.6 TERMINOLOGICKÉ FAKTY:

Terminologické fakty slouží k větší a přirozenější provázanosti faktů uvedených v bázi znalostí. Pokud jsme v naší bázi uvedli, že je Karel muž, chtěli bychom na dotaz „Žena(Karel)“ dostat odpověď **ne**. Nebo pokud jsme v naší bázi znalostí uvedli, že Eliška a Viktor jsou manželé větou ŽenatýS[*viktor,eliška*], chtěli bychom, aby výsledkem dotazu ŽenatýS[*eliška,viktor*] byla také **true**. V naší bázi znalostí ovšem takové usuzování dosud nemělo podporu. A úkolem terminologických faktů je právě množina vět, která by podporovala dedukci na základě terminologie, jakou používáme. Zde si uvedeme několik příkladů:

Neslučitelnost:

Často si 2 predikáty odporují a z předpokladu jednoho musí vyplývat negace druhého

$$\forall x[\text{Muž}(x) \supset \neg \text{Žena}(x)]$$

Podmnožina:

Spousta predikátů označuje druh specializace, kde jeden typ je zahrnut v jiném. Řekli bychom, že je jeho podmnožinou. Například chirurg je druh doktora, což bychom zapsali jako

$$\forall x[\text{Chirurg}(x) \supset \text{Doktor}(x)]$$

Tím zajistíme, že cokoliv bude platit o doktorech, bude platit zároveň I o chirurzích. (Ovšem ne obráceně! To, že se nějaký fakt vztahuje na chirurgy ještě neznamená, že je platný pro všechny doktory).

Úplnost:

De-facto opak výroku podmnožin. Kompletní výčet podtypů (dva nebo více) se složí, aby vytvořili nadmnožinu:

$$\forall x[\text{Dospělý}(x) \supset (\text{Muž}(x) \vee \text{Žena}(x))]$$

Symetrie:

Tak jako už zmíněný případ s manželi, existují symetrické vztahy

$$\forall x,y[\text{ŽenatýS}(viktor,eliška) \supset \text{ŽenatýS}(eliška, viktor)]$$

Inverze:

Některé vztahy jsou v opačném poměru

$$\forall x,y[\text{Potomek}(x,y) \supset \text{Rodič}(y,x)]$$

Omezení:

Význam některých predikátů s sebou nese i jistá omezení, která musíme v jazyce FOL definovat. Například u manželství se předpokládá, že partnery jsou osoby. V tomto případě se jedná o omezení typu predikátů:

$$\forall x,y[\text{ŽenatýS}(x,y) \supset \text{Osoba}(x) \wedge \text{Osoba}(y)]$$

Nové definice:

V některých případech je pro nás vhodné sloučit predikáty již definované v bázi znalostí a vytvořit nový predikát, jako jejich logickou kombinaci

$$\forall x[\text{BohatýMuž}(x) \equiv \text{Bohatý}(x) \wedge \text{Muž}(x)]$$

Jak můžeme vidět u všech použitých příkladů, terminologické fakty bývají zapsány v jazyce FOL jako univerzálně kvantifikované podmínky.

4. ALGORITMIZACE VÝRAZŮ FOL

Viz. [3]

Pro potřeby následujícího textu bude na úvod uvedena trojice důležitých pojmů.

Formule A je logicky platná neboli tautologie, platí-li I(A[v]) = true při každé valuaci v všech výrokových proměnných

---

Formule A je splnitelná, existuje-li valuace v všech výrokových proměnných, při níž I(A[v]) = true

---

Formule A je nesplnitelná, neboli kontradikce, není-li splnitelná

Ještě by bylo dobře připomenout, že valuací výrokových proměnných je myšleno přiřazení určité pravdivostní struktury výrokovým proměnným, reprezentujícím elementární výroky o modelovaném světě.

Algoritmizací výrazů FOL je myšlen jejich přepis do takového tvaru, kdy se za určitých okolností dá o formuli prohlásit, zda je či není splnitelná, bez jakékoliv vazby na význam proměnných, které jsou ve formuli obsaženy.
K tomu je ovšem třeba převést danou formuli do tzv. klauzulární formy.

Přímým způsobem je formule postupnými kroky odvozována z výchozích předpokladů, pomocí daných pravidel.
Nepřímé postupy naopak spočívají v tom, že formule, která má být z daných předpokladů dokázána je popřena, z čehož pak postupnými kroky odvozování

vyplyne spor. Při důkazu logické platnosti formule postupují nepřímé důkazové metody tak, že dokazují nesplnitelnost negace této formule.

U rezolučního důkazu se předpokládá, že formule je logickým důsledkem dané množiny předpokladů, právě když je množina formulí sestávající ze všech daných předpokladů a negace předpokládaného závěru nesplnitelná.

Rezoluční odvozování slouží ke zjišťování splnitelnosti formulí, jejich logické platnosti, resp. logických důsledků množin formulí.

4.1 PRENEXNÍ KLAUZULÁRNÍ FORMA

K tomu, aby se dalo využít výhod rezolučního odvozování, je třeba transformovat formule predikátové logiky do takzvané prenexní klauzulární formy.

K převodu se využívá řady logicky platných ekvivalencí, které bývají velmi užitečnými prostředky.

a(x) a b(x) jsou libovolné formule a, b jsou formule, v nichž se x nevyskytuje volně:

(1) $\models \forall x\ (a(x) \,\&\, b(x)) \Leftrightarrow (\forall x\ a(x) \,\&\, b(x))$

(2) $\models \forall x\ (a(x) \vee \forall x\ b(x)) \Rightarrow \forall x\ (a(x) \vee b(x))$

(3) $\models \exists x\ a(x) \vee b(x) \Leftrightarrow (\exists x\ a(x) \vee \exists x\ b(x))$

(4) $\models \exists x\ a(x) \,\&\, b(x) \Rightarrow (\exists x\ a(x) \,\&\, \exists x\ b(x))$

(5) $\models (\forall x\ a(x) \,\&\, b) \Leftrightarrow \forall x\ (a(x) \,\&\, b)$

(6) $\models (\forall x\ a(x) \vee b) \Leftrightarrow \forall x\ (a(x) \vee b)$

(7) $\models (\exists x\ a(x) \,\&\, b) \Leftrightarrow \exists x\ (a(x) \,\&\, b)$

(8) $\models (\exists x\ a(x) \vee b) \Leftrightarrow \exists x\ (a(x) \vee b)$

(9) $\models \forall x\ (a(x) \rightarrow b) \Leftrightarrow \exists x\ (a(x) \rightarrow b)$

(10) $\models \exists x\ (a(x) \rightarrow b) \Leftrightarrow \forall x\ (a(x) \rightarrow b)$

(11) $\models \forall x\ (a \rightarrow b(x)) \Leftrightarrow (a \rightarrow \forall x\ b(x))$

(12) $\models \exists x\ (a \rightarrow b(x)) \Leftrightarrow (a \rightarrow \exists x\ b(x))$

4.2 PŘEVOD FORMULE DO PRENEXNÍ NORMÁLNÍ FORMY

V prvním kroku se všechny kvantifikátory soustředí před formuli do prefixu formule.

Formule A je v prenexní normální formě, pokud má tvar $Q_1x_1 \ldots Q_nx_n\ B$, $n>0$, *přičemž:*

1) Pro každé i = 1,2, … n je Q_i buď $\forall$ nebo $\exists$
2) $x_1, x_2, \ldots x_n$ jsou navzájem různé proměnné
3) B je otevřená formule

Formule B se pak nazývá otevřené jádro (matice) formule A a posloupnost $Q_1x_1 \ldots Q_nx_n$ prefix formule A.

Abychom formuli dostali do prenexního tvaru, použijeme prenexních operací, což jsou v podstatě přepisovací pravidla z jednoho tvaru na druhý.
Jedná se o následující operace:

Standardizace vázaných proměnných, spočívající v jejich přejmenování, tak aby se předsunutím symbolu kvantifikace do prefixu formule nezměnily meze jeho působnosti
Odstranění nadbytečných kvantifikátorů, ke kterým neexistují příslušné vázané proměnné.
Přesun spojky negace umístěné před závorkou nebo před kvantifikátorem před atom podle následujících schémat:

$\neg\neg$ A	……	A
$\neg\forall x$ A	……	$\exists x \neg$A
$\neg\exists x$ A	……	$\forall x \neg$A

$\neg(A \,\&\, B)$ ..... $(\neg A \vee \neg B)$

$\neg(A \vee B)$ ..... $(\neg A \,\&\, \neg B)$

Přesun kvantifikátorů před formuli pomocí přepisovatelných pravidel (a) – (e):

a) $Q_x\, A(x) \bullet Q_x\, B(x)$ _ _ _ _ _ _ _ _ _ _ _ $Q_x\,(A(x) \bullet B(x))$

b) $Q_x\, A(x) \bullet B$ _ _ _ _ _ _ _ _ _ _ _ $Q_x\,(A(x) \bullet B)$

c) $A \bullet Q_x\, B(x)$ _ _ _ _ _ _ _ _ _ _ _ $Q_x\,(A \bullet B(x))$

d) $Qx\, A(x) \rightarrow B$ _ _ _ _ _ _ _ _ _ _ _ $Q'_x\,(A(x) \rightarrow B)$

e) $B \rightarrow Qx\, A(x)$ _ _ _ _ _ _ _ _ _ _ _ $Q_x\,(B \rightarrow A(x))$

Kde symbol „•“ zastupuje & nebo $\vee$, symbol „Q“ zastupuje $\exists$ nebo $\forall$, A, B jsou formule. Spojka Q’ představuje opačný kvantifikátor od Q.

Z prenexní formule lze dalším postupem vytvořit klauzulární formu. V klauzulární formě se nevyskytují žádné volné proměnné a prefix formule tvoří pouze univerzální kvantifikátory. Pro klauzulární formy formulí už existuje řada algoritmů rozhodování splnitelnosti, resp. logické platnosti formule.

Formule A predikátové logiky je uzavřenou formulí v prenexní klauzulární formě, jestliže má tvar $\forall x_1, \ldots \forall x_n$, B, kde

- $x_1, \ldots x_n$ jsou všechny proměnné, které se ve formuli vyskytují
- B je otevřená formule v klauzulární formě.

Protože převod formule do její klauzulární formy má za cíl možnost další manipulace pouze s otevřeným jádrem formule, o němž se předpokládá, že všechny jeho proměnné jsou univerzálně vázány, nelze žádnou z proměnných ponechat před úpravou volnou. Formule je proto do klauzulárního tvaru upravována v těchto krocích:

a) Změna volných proměnných na vázané zavedením existenčních kvantifikátorů

b) Transformace formule do ekvivalentní prenexní formy
c) Transformace otevřeného jádra formule do klauzulární formy
d) Provedení skolemizace, která eliminuje existenční kvantifikátory

Princip skolemizace spočívá v úvaze, že výraz $\forall x \exists y\ f(x,y)$, značí funkční vztah mezi x a y tak, že x v náhradní funkci značí nezávisle proměnnou a y na ní závisí.
Existenční kvantifikátor lze pak eliminovat pomocí nějaké obecné funkce:
$\forall x_1 \forall x_2 \exists y\ f(x_1,x_2,z)$ nahradíme $\forall x_1 \forall x_2\ f(x_1,x_2,g(x_1,x_2))$

Náhrada existenčního kvantifikátoru v prefixu formule A v klauzulární prenexní forma pomocí Skolemova funktoru zachovává splnitelnost této formule.

4.2.1 Splnitelnost:

Obecně aplikovatelný algoritmus rozhodování splnitelnosti či platnosti neexistuje, nicméně stále platí, že výrazy můžeme interpretovat pouze jako true nebo false a zároveň platí, že jejich negace musí být interpretována opačnou hodnotou.
Jedním z nejefektivnějších formálních způsobů, jak prověřit, zda je daná formule logickým důsledkem dané množiny formulí, je rezoluční formální důkaz.

4.3 REZOLUČNÍ FORMÁLNÍ DŮKAZY

Před použitím rezolučního odvozovacího pravidla je třeba dvojici pozitivního a negativního literálu příslušného atomu ve dvou různých klauzulích vhodnými substitucemi unifikovat, tak, aby navzájem sobě odpovídající termy (pozicí v seznamu termů) vyskytující se v predikátu, byly shodné.

Proces provedení nutných substitucí termů se nazývá *unifikace dvojice atomů*.
Substituce termů $\{t_1/x_1 \ldots t_n/x_n\}$ se nazývá jejich unifikátorem.

Nalezení unifikantů není vždy jednoduché. K pomoci může řešení pomocí množiny neshod, která tento problém převádí na řešení soustavy rovnosti mezi termy na stejných pozicích.

4.3.1 Příklad nepřímého rezolučního důkazu na příkladu 3.13:

Předpokládejme, že máme 3 barevné kostky poskládané jako věž, jednu na druhé. O nejvyšší kostce víme, že je zelená, o spodní kostce víme, že není zelená a o prostřední nemáme vůbec žádné informace. Otázka zní, jestli se v naší věžičce vyskytuje zelená kostka ležící přímo na nezelené.

Predikát **N** určuje relaci kostek položených na sobě.
Predikát **Z** značí zelenou barvu.

1)	N(a,b)	
2)	N(b,c,)	
3)	Z(a)	
4)	¬Z(c)	
5)	¬Z(x) ∨¬N(x,y) ∨Z(y)	klauzulární forma popření
6)	¬Z(a) ∨¬N(a,y) ∨Z(y)	unifikace 5, 3
7)	¬N(a,y) ∨ Z(y)	rezoluce 6, 3
8)	¬N(a,b) ∨ Z(b)	unifikace 7, 1
9)	Z(b)	rezoluce 1, 8
10)	¬Z(b) ∨¬N(b,y) ∨Z(y)	unifikace 5, 9
11)	¬N(b,y) ∨ Z(y)	rezoluce 9, 10
12)	¬N(b,c) ∨ Z(c)	unifikace 11, 2
13)	Z(c)	spor 4 a 13
14)	X	

4.3.2 Shrnutí

Klauzulární prenexní forma formulí predikátové logiky je výhodným tvarem, protože na ni lze aplikovat formální důkazy splnitelnosti formulí predikátové logiky.

Ty formule, které jsou v nějakém obecnějším tvaru je třeba do klauzulární prenexní formy převést. Některé počítačové systémy pracující s predikátovou logikou pracují výhradně s tímto tvarem formulí.

5. NEVÝHODY PREDIKÁTOVÉ LOGIKY

Za největší nevýhodu logiky prvního řádu lze považovat její schopnost vyjádřit pouze dva pravdivostní stavy, false a true. O jakémkoliv prvku, či vlastnosti jsme schopni uvést pouze to, zdali patří nebo nepatří do nějaké námi určené množiny XY.

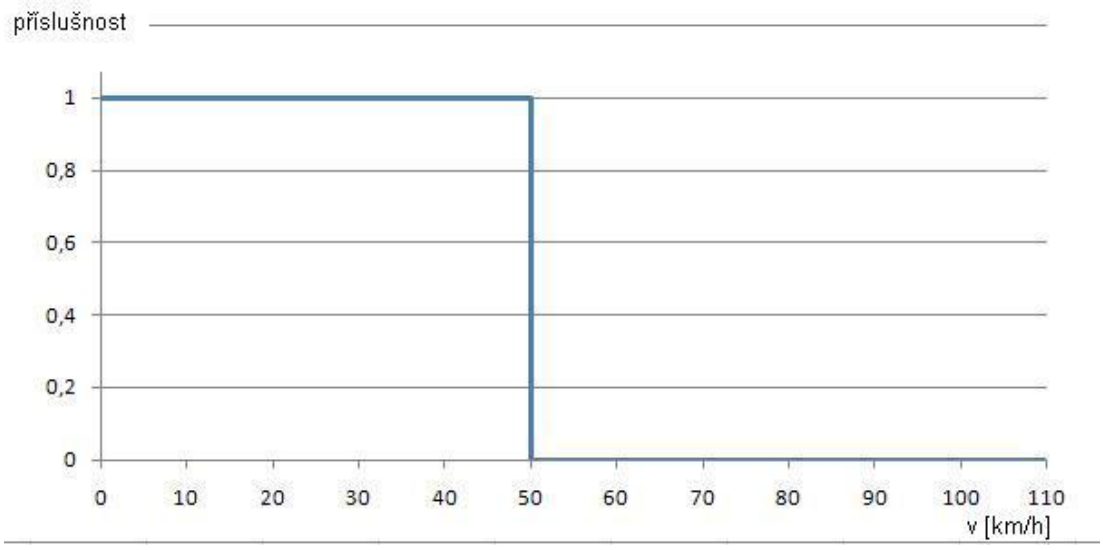
Například zákon č. 361/2000 sb. § 18 odst. 4 uvádí:

> ***V obci smí jet řidič rychlostí nejvýše 50 km/h, a jde-li o dálnici nebo silnici pro motorová vozidla, nejvýše 80km/h.***

Pokud tedy jedeme v obci po veřejné komunikaci, která není klasifikována jako dálnice či silnice pro motorová vozidla rychlostí 50 km/h, je všechno v pořádku (uvážíme-li, že na komunikaci není žádné místní omezení).

Pokud se ručička tachometru našeho vozidla přehoupne přes 50 km/h, dopouštíme se přestupku, za který můžeme být pokutováni.

Jedná se o určitou absurdnost, provázející „ostré hrany“. Pro určité hodnoty funkce je jistý výraz vyhodnocen jako true, kdežto další hodnota, vyšší, resp. nižší o nekonečně malý přírůstek už je vyhodnocena jako false. Grafické znázornění příkladu s rychlostí auta v obci, by mohlo vypadat následovně:



Obrázek 4: funkce příslušnosti prvků do množin pro predikátovou logiku

Kdy pro rychlosti do 50 km/h se rychlost našeho vozidla pohybuje v povolených mezích (na obrázku vyjádřeno stavem 1), kdežto při rychlosti, byť o nejnepatrnější přírůstek vyšší, už se vyskytujeme za hranou povolené rychlosti a tedy i zákona (na obrázku vyjádřeno stavem 0).

Omezení v rozsahu hodnot se dá v jistých případech kompenzovat například nastavením hranic množin tak, abychom měli jistotu, že nevynecháme žádný relevantní prvek, ovšem ve většině případů je tento problém nepřekonatelný.

Pokud se budeme pomocí predikátové logiky snažit modelovat realitu, setkáváme se s její přílišnou složitostí, kterou nejsme schopni vystihnout a s neurčitostí, která je způsobena naší neschopností přesně realitu diferencovat. Matematický popis vyžaduje velkou přesnost a to vynucuje řadu zjednodušení. Tím se model stává nevýstižným. Použití přesných pojmů, které vyžaduje dvouhodnotová logika, je tedy aplikovatelné pouze v ideálním případě.

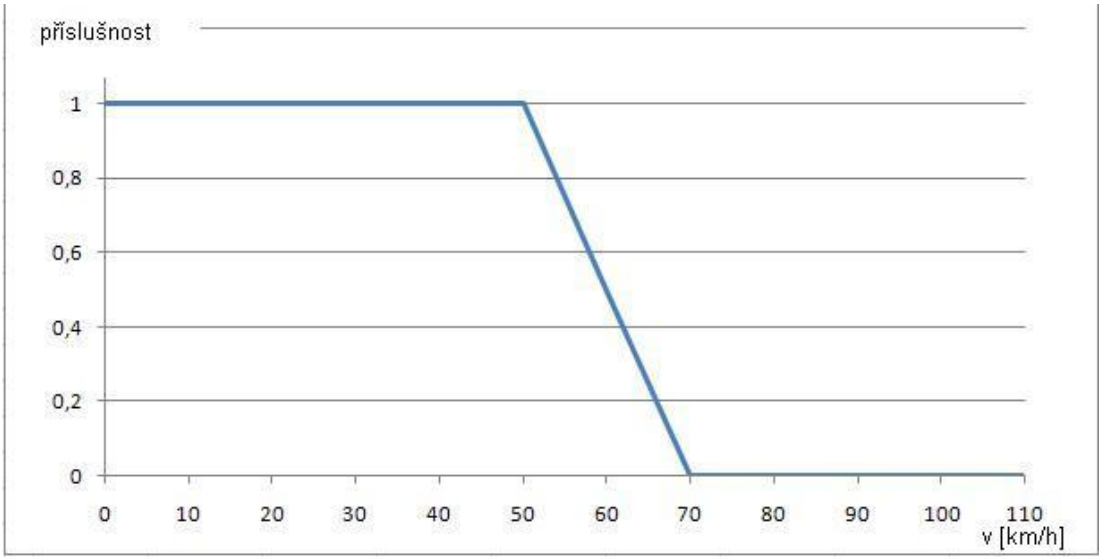
5.1.1 Fuzzy přístup:

Vyřešení problému dvouhodnotového vyjádření nabízí například fuzzy logika. Prvek fuzzy množiny se skládá ze dvou atributů. První atribut vyjadřuje hodnotu prvku samotného. Druhý atribut vyjadřuje míru příslušnosti k dané množině. Pokud se tedy přidržíme příkladu s maximální povolenou rychlostí, mohly by prvky fuzzy množiny „přiměřená_rychlost“ vypadat nějak takto:

rychlost	příslušnost
km/h	[-]
49	1
50	1
52	0,9
56	0,7
60	0,5
64	0,3
68	0,1
70	0
71	0

Tabulka 1: Hodnoty prvků fuzzy množiny

Grafické vyjádření fuzzy množin:



Obrázek 5: funkce příslušnosti prvků do množin pro fuzzy logiku

Pojmenujme negaci množiny „přiměřená_rychlost“ pojmem „nepřiměřená_rychlost“

$$\neg(\textit{přiměřená\_rychlost}) = (\textit{nepřiměřená\_rychlost})$$

Prvky v rozmezí 50 – 70 km/h jsou členy jak množiny přiměřených rychlostí (s uvedenou příslušností), tak množiny nepřiměřených rychlostí (s příslušností doplňku do jedné). Tato interpretace velmi přesně odpovídá neurčitým vyjádřením, která jsou nám, lidem, tak blízká. V reálném životě posuzujeme vlastnosti objektů výrazy typu „hodně“, „málo“, „mladý“, „asi“ apod., přičemž jednotlivým slovnímu výrazu odpovídá jiný tvar a sklon relační křivky, kterou navíc každý jedinec vnímá jinak. Člověk, kterému je 70 let zcela jistě vnímá výraz „starší osoba“ odlišně, než dvacetiletý jedinec.

Co se logiky prvního řádu týče, je výše uvedená interpretace prvků zcela nemožná. Ba právě naopak. Celé odvozování pomocí predikátové logiky je postaveno na tom, že prvek, vyhovující relaci p(x), nemůže vyhovovat relaci ¬p(x).

5.1.2 Bayesovský přístup

Další možností, je použít systém založené na bayesovské pravděpodobnosti se spojitým oceňováním antecedentů i konsekventů pravidel hodnotami z intervalu <0;1>. Jako příklad použiji expertní systém vyvinutý na VUT Brno, NPS32.

Tento systém pracuje na principu ohodnocování jednotlivých uzlů pravděpodobnostními hodnotami z rozsahu 0 – 99%. (hodnota 100 nebyla použita z důvodu inferenčních pravidel, kde by se vyskytovalo dělení nulou).
Znalostní inženýr může při tvorbě báze znalostí různým ohodnocením uzlu vyjádřit míru důležitosti odpovědi pro daný uzel.
Navíc už není uživatel nucen vybírat pouze ze dvou pravdivostních hodnot. Počet možných odpovědí stejně jako jejich pravdivostní ohodnocení určuje opět znalostní inženýr (dle požadavků na správné vyladění báze znalostí).

Jednoduchá báze na výběr vhodného restauračního zařízení by mohla vypadat následovně:

```
;.Báze znalostí k výběru vhodného restauračního zařízení
;?5
.Fotbalek p=50% D
;Měl by být v restauraci přítomen stolní fotbal?
.Kulecnik p=50% D
;Měl by být v restauraci přítomen kulečník?
.Vareni   p=50% D
;Měli by v restauraci vařit teplá jídla?
.uvy      p=30% G
;U Vycepu
& 60%    +Fotbalek
& 50%    +Kulecnik
& 80%    +Vareni
.utr      p=30% G
;U Troje
& 70%    -Fotbalek
& 70%    -Kulecnik
& 70%    -Vareni
.ubs      p=30% G
;U Busku
& 80%    +Fotbalek
& 70%    -Kulecnik
& 60%    +Vareni
.upl      p=30% G
;U Palecka
& 70%    -Fotbalek
& 70%    -Kulecnik
& 70%    +Vareni
#
```

Charakteristiky koncových uzlů (hypotéz), neboli restauračních zařízení, ze kterých vybíráme, jsou následující:

Restaurace U Výčepu:	Stolní fotbal:	ano
	Kulečník:	ano
	Teplá jídla:	ano
Restaurace U Tróje:	Stolní fotbal:	ne
	Kulečník:	ne
	Teplá jídla:	ne
Restaurace U Bušků:	Stolní fotbal:	ano
	Kulečník:	ne
	Teplá jídla:	ano
Restaurace U Palečka:	Stolní fotbal:	ne
	Kulečník:	ne
	Teplá jídla:	ano

Ti, kteří se s Expertním systémem NPS setkali, jistě potvrdí, že uvedená báze znalostí patří mezi nejjednodušší možné. Provede uživatele sérií 3 dotazů, na které mají možnost vybírat odpověď z pěti standardních možností a to:

Určitě ano	*(100%)*
Snad ano	*(66,7%)*
Nevím	*(50%)*
Asi ne	*(33,3%)*
Určitě ne	*(0%)*

Odpovědi vyjadřují míru jejich ztotožnění se s předmětem otázky.

Na rozdíl od predikátové logiky tedy může uživatel vyjádřit míru toho, jak moc se má jeho odpověď promítnout do výsledku. Pokud na dotaz:

Měli by v restauraci vařit teplá jídla?

Odpoví „*Snad ano*“, vyjadřuje tím skutečnost, že toto kritérium výběru pro něj není nijak zásadní, nicméně je vítaným prvkem.

V predikátové logice by si musel vybrat mezi stavy „*Určitě ano*“ a „*Určitě ne*“ přičemž jeho volba by nezvratně ovlivnila výsledná data, jak bude ukázáno v následujícím textu.

Projděme si konzultací systému NPS za použití výše uvedené báze znalostí s následujícími odpověďmi:

Měl by být v restauraci přítomen stolní fotbal? *Určitě ano*
Měl by být v restauraci přítomen kulečník? *Určitě ne*
Měli by v restauraci vařit teplá jídla? *Určitě ne*

Výsledkem konzultace je následující tabulka:

Tabulka 2: Výsledek konzultace

pořadí	Název	Pravděpodobnost
1	U Bušků	74,1
2	U Tróje	58,8
3	U Palečka	11,4
4	U Výčepu	9,7

Ač to na první pohled nejspíš nebylo zřejmé, výběr odpovědí nebyl náhodný.

Záměrně byly vybrány takové odpovědi, kterým plně nevyhovuje žádná z nabízených možností.

Systém typu NPS32 dokáže vybrat nejrelevantnější odpověď ze všech možných, respektive setřídit všechny hypotézy dle relevantnosti.

U systému založeného na predikátové logice toto není možné, protože dotaz (vzorec), který by hledal vyhovující odpověď v zadaném universu diskurzu (obsahujícím predikáty restaurací a funktory přiřazující jednotlivým restauracím stolní fotbaly, kulečníky a teplou kuchyni jako jejich vlastnosti) by nemohl žádnou najít. Jevil by se jako nesplnitelná formule.

5.1.3 Shrnutí

Použití pokročilejších metod zvládajících práci s neurčitostí dává mnohem citlivější výsledky na požadavky uživatele, než při použití predikátové logiky, která používá striktní označení „true“ a „false“ pro hodnocení svých formulí.

6. ZÁVĚR

Predikátová logika je vhodná pro úlohy typu „výběr jediné možnosti z mnoha“. Zcela jistě jí nelze upřít její vyjadřovací sílu, nicméně jen málo reálných úloh lze idealizovat tak, abychom je byli schopni vyřešit pouze za pomoci logiky prvního řádu.

Většinou se jedná o úlohy typu doplnění prvků do relací, aneb „výběr jedné možnosti z mnoha“, popřípadě výběr všech možných řešení. Jedním z typických příkladů, které je schopna predikátová logika řešit samostatně jsou například úlohy typu „hanojské věže“ apod.

V Expertních systémech, které využívají pokročilejší metody inference, může však být predikátová logika stále velmi silný nástroj, vhodný například k odvozování implicitních znalostí, tj. těch vztahů, které nejsou dány explicitně.

7. SEZNAM POUŽITÉ LITERATURY

[1] BRACHMAN, J., LEVESQUE, H. ***KNOWLEDGE REPRESENTATION AND REASONING,*** ELSEVIER 2004. ISBN 978-1-55860-932-7

[2] ŠTĚPÁN, MAŘÍKOVÁ, AJ. ***UMĚLÁ INTELIGENCE 2,*** ACADEMIA 1997, ISBN 80-200-0504-8

[3] LUKASOVÁ, A. ***FORMÁLNÍ LOGIKA V UMĚLÉ INTELIGENCI,*** COMPUTER PRESS 2003, ISBN 80-251-0023-5

[4] PEREGRIN, J. ***LOGIKA A LOGIKY,*** ACADEMIA 2004, ISBN 80-200-1187-0

[5] ŠVEJDAR, V. ***LOGIKA NEÚPLNOST, SLOŽITOST A NUTNOST,*** ACADEMIA 2002, ISBN 80-200-1005-X

[6] LHOTSKÁ, L. ***ZÁKLADY UMĚLÉ INTELIGENCE [ONLINE].*** **2005. DOSTUPNÉ Z:** HTTP://CYBER.FELK.CVUT.CZ/GERSTNER/TEACHING/ZUI/ZUI.HTML

[7] DVOŘÁK, J. ***EXPERTNÍ SYSTÉMY [ONLINE].*** **2004. DOSTUPNÉ Z:** <HTTP://WWW.UAI.FME.VUTBR.CZ/~JDVORAK/OPORY/EXPERTNISYSTEMY.PDF>

[8] THON, M. ***EXPERTNÍ SYSTÉMY [ONLINE].*** **2006. DOSTUPNÉ Z:** <HTTP://MILOST.WZ.CZ/UMI/REFERAT/KLASIFIKACE.HTML>

[9] RUBÁČEK, F. ***PROLOG [ONLINE].*** **2005. DOSTUPNÉ Z:** HTTP://IRIS.UHK.CZ/LOGPRO/