OSTRAVSKÁ UNIVERZITA V OSTRAVĚ

EXPERTNÍ SYSTÉMY

MIROSLAV POKORNÝ

OSTRAVA 2004

Expertní systémy

Recenzenti:

Název:	Expertní systémy
Autor:	Prof. Dr. Ing. Miroslav Pokorný
Grafická úprava:	Ing. Pavel Petránek
Vydání:	první, 2004
Počet stran:	103
Náklad:	
Tisk:	Ediční středisko CIT OU

Studijní materiály pro distanční kurz: Expertní systémy

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.
Určeno výhradně pro kurzy Celoživotního vzdělávání Moravskoslezska

Vydavatel a tisk: Ostravská univerzita v Ostravě,
Systém celoživotního vzdělávání Moravskoslezska

ISBN

Cíl modulu:

Záměrem tohoto modulu je vysvětlit studentům základní pojmy z oblasti umělé inteligence, která je teoretickým zázemím expertních systémů jako nejrozšířenějšího nástroje tohoto moderního vědního oboru. Expertní systémy jako programové systémy pro počítačovou podporu rozhodování při řešení složitých úloh jsou dnes úspěšně aplikovány v celé řadě technických i netechnických oborů. Modul je věnován problematice metod, které tvoří základy expertních systémů, věnuje se jejich vlastnostem i funkcím

Modul uvádí zázemí změny paradigmatu, kterým se řídí zažité stereotypy našeho myšlení. V praktické oblasti rozvoje počítačových věd jde především o přechod od zpracování dat ke zpracování znalostí, tedy informací, které jsou základem efektivního myšlení a dosahování jeho vysoce kvalitních výsledků a závěrů.

Jednotlivé kapitoly modulu objasňují teoretické principy, o něž se opírá řešení problematiky základních funkcí expertních systémů, především efektivní počítačové reprezentace znalostí a dále procedur, které nad těmito znalostmi operují a efektivně je využívají k řešení složitých úloh z různých problémových oblastí. Zvýšená pozornost je věnována té části látky, která umožní studentům nejen expertní systémy pochopit, ale i samostatně navrhovat a programově realizovat. Studenti jsou seznámeni se základy programovacího jazyka umělé inteligence PROLOG, větší pozornost je věnována expertním systémům využívajících fuzzy množinové matematiky a fuzzy logiky. Důležité části látky jsou ilustrovány vhodnými příklady.

Globální cíl studia modulu bude splněn pomocí cílů dílčích stanovených v úvodu jednotlivých kapitol. K jejich zvládnutí studentům pomohou kurzívou psané části označené ikonou kompasu nazvané Průvodce studiem, kde se dozvědí, kolik času budou přibližně potřebovat na prostudování jednotlivých kapitol

Modul obsahuje vybrané metody a z nich vyplývající techniky pro tvorbu expertních systémů (logické programování, fuzzy-logické přístupy) a uvádí příklady jejich použití. Poskytuje studentům základní informace a připravuje je tak k eventuálnímu zvládnutí problematiky návrhu, tvorby a ladění expertních systémů v plném rozsahu jejich dalším studiem.

Slušelo by se zde na úvod uvést i celkový čas potřebný ke zvládnutí celého modulu. Sečtením časových požadavků jednotlivých kapitol můžete zjistit, že celý modul lze zvládnout přibližně za 23 hodin studia. Je však třeba to považovat za údaj s nízkou vypovídací hodnotou. Je nutné brát v úvahu, že pro některé studenty bude látka náročná a nelze tedy plánovat zvládnutí např. jedné kapitoly denně, natož pak více.

Za optimální bych v takovém případě považoval rozvržení studia nejméně do 5 týdnů, pokud možno i do delšího období více, aby byl čas na přestávky umožňující lepší promyšlení látky.

Kurs by měl být vždy absolvován za podpory tutora, který je dostatečně erudovaný v oboru umělé inteligence a který bude nejen hodnotit korespondenční úkoly, ale také bude studentům nápomocen při porozumění obtížnějším pasážím problematiky.

Přeji všem čtenářům, aby předložená studijní podpora jejich studium problematiky expertních systémů usnadnila a aby měli brzy radost z dosažených výsledků.

Miroslav Pokorný

OBSAH

1 ÚVOD DO EXPERTNÍCH SYSTÉMŮ

V této kapitole se dozvíte:

- Co je umělá inteligence a jaký je její vztah k inteligenci lidské
- Něco o historickém vývoji vědního oboru umělá inteligence
- Jaký je vztah umělé inteligence a objektu našeho hlavního zájmu – expertních systémů

Po jejím prostudování byste měli být schopni:

- Vysvětlit pojem umělé inteligence a její náplň jako vědního oboru
- Pojednat o hlavních etapách jejího vývoje a nejvýznamnějších výsledcích
- Charakterizovat význam a princip činnosti expertních systémů

Klíčová slova této kapitoly:

Lidská inteligence, umělá inteligence, znalosti objektivní, znalosti subjektivní, zpracování znalostí, expertní systém

Doba potřebná ke studiu: 3 hodiny

Průvodce studiem

Studium této kapitoly může být poměrně náročné zejména pro ty z Vás, kteří se dosud s problematikou teorie a použitím praktických nástrojů umělé inteligence nesetkali. V takovém případě Vám zřejmě některé pojmy budou připadat obtížně pochopitelné, ovšem nenechte se tím odradit, potřebné souvislosti vyplynou z látky probírané v dalších kapitolách.
Na studium všech čtyř částí této úvodní kapitoly pojednávající o základních přístupech umělé inteligence a jejím vztahu k expertním systémům si vyhraďte zhruba 2 hodiny. Její zvládnutí včetně odpovědí na korespondenční otázky by Vám nemělo trvat déle než 3 hodiny, někteří jej určitě zvládnou i za dobu kratší.

1.1 Umělá inteligence

Motto:
Krach může mít tři příčiny: ženy, sázky nebo rady expertů.
Georges Pompidou

Předběhněme dějiny, usmějme se nad úvodním filozofickým bonmotem pana Pompidoua a uveďme hned na počátku textů studijní pomůcky, kterou jste právě vzali do ruky, stěžejní definici fenoménu z jejího názvu – definici expertního systému. Použijme základ myšlenky E.A.Feigenbauma ze Stanfordské univerzity v Kalifornii (1988) [1] a konstatujme, že

„Expertní systémy jsou počítačové programy, simulující rozhodovací činnost člověka (experta) při řešení složitých úloh s podstatným cílem - dosáhnout kvality rozhodnutí na jeho úrovni“.

Tato definice bezprostředně evokuje otázku co je to vlastně kvalita rozhodnutí a čím je dána? U člověka je odpověď nasnadě – kvalita rozhodnutí je dána komplexem jeho mentálních schopností, které řadíme do pojmu jeho **lidské inteligence**. Budeme-li pak požadovat od stroje (počítače), aby při řešení stejné situace poskytl stejně (přibližně stejně) kvalitní rozhodnutí, pak musíme zřejmě transponovat pojem inteligence lidské na pojem inteligence stroje, tedy **inteligence umělé**. Intuitivně tušíme, že pojednání o inteligentně se chovajících strojích (počítačích) je nutno zahájit úvahami, které souvisí s naším chápáním inteligence jako takové. Cítíme, že se dostáváme částečně na pole filozofie a musíme sáhnout do historie – i když v našem případě nikterak hluboké.

Umělá inteligence – shrnutí

Expertní systémy jsou specializované počítačové programy, které simulují rozhodovací činnost člověka při řešení složitých problémů. Kvalita lidských rozhodnutí je závislá na úrovni jeho intelektu – komplexu jeho mentálních schopností. Rozhodovací činnosti počítače, pokud mají být kvalitní, musí vykazovat specifické vlastnosti svých procedur, které jsou řešeny v rámci vědního oboru umělá inteligence.

Kontrolní otázky:

1. Jak chápeme pojem přirozené a umělé inteligence?

Otázka k zamyšlení:

1. Jak se vytváří v průběhu vývoje člověka komplex jeho inteligentních vlastností?

Úkoly k zamyšlení:

Promyslete druhy a význam inteligentních vlastností potřebných pro vykonávání Vašeho povolání.

1.2 Trocha historie úvodem

Dnes můžeme říci, že umělá inteligence je jednou z nejrychleji se rozvíjejících vědních disciplín v celé historii. Až v létě roku 1956 byla totiž na Darmtouth College v New Hampshire (USA) zorganizována poměrně malá konference „The Darmouth Summer Research Project on Artificial Intelligence“, na kterou byli pozváni přední odborníci zabývající se mentálními schopnostmi lidí a strojů. Sešli se tak odborníci z oblasti matematiky, elektrotechniky, elektroniky, lingvistiky, neurologie, psychologie a filozofie. Tato konference se zapsala do dějin umělé inteligence zlatým písmem. Jejím náplní byla diskuze domněnky, že

„Každé hledisko učení nebo jakýkoliv jiný příznak inteligence může být v principu tak přesně popsán, že může být vyvinut stroj, který ho simuluje“ [2].

Byla poprvé vyslovena myšlenka, že počítače by mohly pracovat se symboly stejně dobře jako s čísly. Díky návrhu organizátora konference – J. McCarthyho – byla nově se rýsující vědní disciplína nazvána – již v názvu konference - **umělou inteligencí**.
Představy odborníků v polovině padesátých let byly velmi optimistické. Vědci předpovídali, že do 25 let nahradí počítače veškerou lidskou intelektuální činnost. Bohužel, historie vývoje umělé inteligence jim za pravdu nedala.
Celá – byť relativně krátká – historie rozvoje umělé inteligence je velmi zajímavá, vymyká se však poněkud poslání této učebnice. Zájemce může nalézt podrobnosti a hlavně literární odkazy v [3]. V krátkosti lze říci, že historický vývoj umělé inteligence prošel třemi základními etapami.

První etapa je datována do zbytku 50. a celých 60. let minulého století. Je typická nadšeným úsilím odborníků a rozvojem prvních nástrojů umělé inteligence. Především je třeba jmenovat LISP (1960) – dodnes jeden z nejpoužívanějších jazyků umělé inteligence a dále systém GPS – General Problem Solving (1960) – nejznámější program pro řešení obecných problémů. Přes řadu úspěšných výsledků se již koncem 60. let začalo ukazovat, že původní optimistické předpovědi byly značně přemrštěné a řešení problémů umělé inteligence bude spojeno s celou řadou problémů.

Druhá etapa, situovaná do 70. let minulého století, je označována jako etapa skepse a stagnace. Bylo omezeno či dokonce zrušeno financování celé řady projektů a omezen výzkum umělé inteligence. Přece však úsilí nadšenců přineslo i v této situaci pozitivní výsledky, soustředěné hlavně do oblasti reprezentace znalostí, prohledávacích technik, zpracování přirozeného jazyka a počítačového vidění. Objevily se první **expertní systémy,** které se staly prototypy dodnes používaných technik (DENDRAL-1971, MYCIN-1976, PROSPECTOR-1978). Byl publikován dodnes široce používaný jazyk umělé inteligence PROLOG (1975).

Třetí etapa z let 80. navázala na výsledky etapy druhé. Je považována za období nového rozvoje umělé inteligence, charakterizované především komercionalizací vyvinutých nástrojů, vytvoření odborné terminologie, publikací monografií k jednotlivým oblastem umělé inteligence, návrhy počítačů s architekturami vhodnými pro umělou inteligenci (LISPovské a PROLOGovské počítače). Byly zahájeny rozsáhlé výzkumné programy, především japonský projekt počítačů páté generace (1981-1990).

Z hlediska dnešního pohledu je místo umělé inteligence v moderní vědě, společnosti i praxi zcela pevné. Její výzkum se stal systematickou součástí odborných pracovišť univerzit a akademií, její produkty jsou postupně rozšiřovány jak v průmyslové tak i ve spotřební elektronice. Uplatňují se v celé řadě technických i netechnických oborů.

Trocha historie úvodem – shrnutí

Historický vývoj umělé inteligence prošel třemi etapami. První představovala nadšený vstup do výzkumu a vývoje specializovaných prostředků, které měly být praktickými nástroji umělé inteligence (programovací jazyk LISP, prostředek pro řešení problémů GPS). Druhá etapa přinesla skepsi a stagnaci, přesto vyprodukovala některé užitečné výstupy (programovací jazyk PROLOG, první použitelné expertní systémy). Třetí etapa

přinesla opět oživení zájmu, hlavně díky dlouhodobému japonskému projektu počítačů páté generace.

Kontrolní otázky:

1. Kdy a čím je datován vznik vědního oboru umělá inteligence?

Otázka k zamyšlení:

1. Jakými etapami prošel vývoj umělé inteligence do dnešních dnů?

Úkoly k zamyšlení:

Byl vývoj umělé inteligence poznamenán některými politickými a hospodářskými situacemi ve světě?

1.3 Co je to umělá inteligence ?

Pojem inteligence [3] je spjat s některými vlastnostmi vyšších organizmů, zvláště člověka. Stručně lze říci, že inteligentní schopnosti umožňují reagovat na složité projevy životního prostředí a aktivně je využívat ve svůj prospěch, k dosažení svých cílů.

Vlastní pojem „inteligence“ je velmi složitý a nebyl nikdy přesně vymezen. Nicméně existují metody pokoušející se kvantifikovat stupeň lidské inteligence (např. IQ-testy). Každá z nich má však své omezení a žádnou nelze považovat za zcela objektivní. Přesto je chování živých organizmů, které je možno považovat za projev jejich inteligence, natolik zajímavé, že neuniklo pozornosti vědců zkoumajících metody a prostředky, jak takové inteligentní chování strojově napodobit – tj. vytvořit umělé systémy, které by vlastnosti inteligentního chování vykazovaly.

Postupy a algoritmy, které by ve svých důsledcích vedly k jistému napodobení projevů inteligentního chování člověka, se s vývojem poznání a vědy staly předmětem zkoumání nové vědní disciplíny – **umělé inteligence** (1956). (Tento termín se postupně zcela vžil a odsunul do pozadí pokusy o jiné interpretace, např. „strojový intelekt“ [4].)
Není-li pojem inteligence u živých organizmů přesně definován, nelze ani čekat přesné vymezení pojmu inteligence umělé. Nehledě na skutečnost, že pokusů bylo v tomto směru provedeno na stovky (např. na VIII. Mezinárodní konferenci IJCAI v roce 1983 v Karlsruhe bylo předloženo na 180 definic umělé inteligence). V současné době převládá názor, že pro vytvoření praktických aplikací nejde ani tak o přesné vymezení pojmu umělá inteligence, jako spíše o vytvoření intuitivní představy o jejím obsahu.

Jestliže expertní systémy ve svých různých variantách patří k dnes nejvíce prakticky rozšířeným nástrojům vědního oboru umělé inteligence, věnujme pozornost alespoň třem definicím umělé inteligence, které můžeme označit jako zdařilé.

Minského definice umělé inteligence

Marvin Minsky [5] předložil v roce 1961 definici, která vychází z tzv. Turingova imitačního testu:

"Umělá inteligence je věda o vytváření strojů a systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby tak postupoval člověk – bychom považovali za projev jeho inteligence."

Hovoří-li definice o řešení určitých úkolů, má na mysli zřejmě úkoly komplikované, složité. Složitost úkolu pak lze ohodnotit počtem všech variant jeho řešení, které připadají v úvahu. Rozhodně nelze považovat za inteligentní způsob řešení postup, při němž jsou postupně přebírány a ověřovány varianty postupně jedna za druhou. Inteligentním naopak nazveme takový postup, při němž budou ověřovány pouze varianty nadějné. Ty, které neposkytují dostatečnou šanci na úspěch řešení, budou apriorně vynechány. Přitom mechanizmus, který umožňuje řešiteli některé varianty apriori odmítnout, je nesporně založen na využívání informací, znalostí o řešeném problému.

Ukazuje se, že **využití znalostí** je pro konstrukci inteligentních postupů zcela relevantní. Znalosti mohou být získány jednak převzetím od člověka, který je schopen úlohu inteligentně řešit (tedy nutně experta v daném oboru), jednak analýzou příkladů a jejich inteligentních řešení.

Lidské znalosti můžeme rozdělit do dvou skupin. Předně jsou to **znalosti objektivní**, obecné či exaktní (někdy nazývané také znalosti hluboké), k nimž mají přístup odborníci např. při studiu daného oboru (teorémy, fyzikální zákony apod.). Druhou skupinu tvoří **znalosti subjektivní**, často heuristické až meta-heuristické (někdy nazývané znalosti mělké – nikoli však ve smyslu povrchní!), získané vlastním poznáním a dlouholetou praxí. Ukazuje se, že pro kvalitu řešení z hlediska umělé inteligence sehrávají tyto subjektivní znalosti podstatnou roli. Zvláštní význam pak budou tyto subjektivní znalosti vykazovat při konstrukci expertních systémů.

Richova definice umělé inteligence

Richova definice umělé inteligence [6] se úspěšně vyhýbá filozofickým úvahám, které většině pokusů o definici inteligence či umělé inteligence dominují. Říká, že

„Umělá inteligence se zabývá tím, jak počítačově řešit úlohy, které dnes zatí lépe zvládají lidé".

Podle tohoto vymezení je obsah umělé inteligence bezprostředně vázán na aktuální stav počítačových věd a lze tedy očekávat, že s rozvojem počítačové techniky se bude těžiště umělé inteligence posouvat a měnit. Jde o velmi stručné a poměrně přesné vymezení toho, co tvoří skutečný obsah umělé inteligence jako vědní disciplíny.

Kotkova definice umělé inteligence

Kotkova definice [7] je blíže spjata s pojmem a posláním technické kybernetiky jako praktické vědy o sdělování a řízení v živých organizmech, strojích a společnosti [8]:

„Umělá inteligence je vlastnost člověkem uměle vytvořených systémů vyznačujících se schopností rozpoznávat předměty, jevy a situace, analyzovat vztahy mezi nimi a tak vytvářet vnitřní modely světa, ve kterých tyto systémy existují, a na tomto základě pak přijímat účelná rozhodnutí, předvídat důsledky těchto rozhodnutí a objevovat nové zákonitosti mezi různými modely nebo jejich skupinami“.

Zavedení definice vnitřních modelů dovoluje definovat postup řešení úlohy takto: je dán model počátečního stavu prostředí a model cílového stavu prostředí. Dále jsou dány přípustné akce, kterými lze stav prostředí měnit. Úkolem je nalézt takové posloupnosti akcí, které převedou počáteční stav do stavu cílového při respektování předem zadaných omezení. Takto formulovaný problém nazýváme v umělé inteligenci **řešením úloh** (jak uvidíme později, vystihuje tato formulace paradigma třídy plánovacích expertních systémů). Formulace modelů a akcí je pak zahrnována do problematiky reprezentace znalostí.

Kotkova definice umělé inteligence umožňuje také explicitně určit a vyjmenovat dílčí teoretické úlohy, které do ní jako vědní disciplíny spadají (rozpoznávání, reprezentace znalostí vč. logiky jako nástroje pro tuto reprezentaci, řešení úloh, adaptace a učení, expertní systémy a komunikace se strojem v přirozeném jazyce – viz kap.2.1). Problematika expertních systémů je pak spjata s problematikou efektivního strojového využívání znalostí špičkových expertů pro obecné úlohy diagnostiky a plánování.

Co je to umělá inteligence - shrnutí

Jediná exaktní definice umělé inteligence neexistuje, z mnoha pokusů lze však vybrat takové, které lze intuitivně označit jako zdařilé. Patří sem především definice Minského (stroje řeší problém postupy, které lze s ohledem na obdobu postupů člověka označit jako inteligentní), definice Richova (počítače zvládají úlohy, dostupné doposud jen člověkovi) a definice Kotkova (definuje pojem vnitřního modelu světa a umožňuje označit technické úlohy, které jsou pro metody umělé inteligence typické).

Kontrolní otázky:

1. Jaké jsou nejznámější definice vědního oboru umělá inteligence?
2. Jaké jsou praktické aspekty Kotkovy definice umělé inteligence?

Otázka k zamyšlení:

1. Absolvoval jste někdy test IQ? Jestliže ano, jak se promítly jeho výsledky do Vašeho dalšího života?

Úkoly k zamyšlení:

Pokuste se definovat umělou inteligenci podle vlastní představy o její náplni.

1.4 Umělá inteligence a expertní systémy

Pro umělou inteligenci jako vědní disciplínu je specifické to, že jde spíše o soubor metod, teoretických přístupů a specializovaných počítačových programů, který sjednocuje úsilí o strojové řešení (strojovou podporu řešení) složitých problémů. Typický je fakt, že dosažením výsledků v řešení problémů přestávají být tyto výsledky součástí umělé inteligence a přecházejí do jiných oborů, kde jsou aplikovány.

Typické jsou příklady použití programových nástrojů aplikovaných v oboru robotiky, moderních metod automatického řízení, informatiky apod.

Expertní systémy, zmíněné hned v úvodu této kapitoly a aplikované dnes v různých oblastech technických i netechnických disciplín, jsou snad nejvýraznějším důkazem využití nástrojů umělé inteligence v praxi. Jsou přirozeným důsledkem poznání, že kvalita systémů s umělou inteligencí závisí daleko více na kvalitě znalostí než na kvalitě mechanizmu pro jejich využívání [9].

Expertní systémy jsou počítačové programy, které simulují rozhodovací činnost expertů při řešení velmi složitých, úzce problémově zaměřených úloh [7]. Je nesporné, že jejich funkce je s lidskou i umělou inteligencí velmi úzce spjata.

Expertní systémy jsou založeny na myšlence převzetí znalostí od experta (tj. jeho znalostí objektivních i subjektivních) a jejich vhodné počítačové reprezentace, která by umožnila počítačovému programu využívat těchto znalostí zhruba stejným způsobem, jako jich využívá expert.

Zajímavé a podstatné je zde hledisko váhy znalostí na kvalitu výsledného rozhodnutí. Znalosti obecné, objektivní, jsou snadněji sdělitelné a lze jich také snadněji nabýt (výuka ve škole). Lze tedy očekávat – a praxe to plně potvrzuje – že dva znalci dané problematiky (experti v daném oboru) budou mít tuto objektivní část znalostí zhruba stejnou a ten z nich, který bude vybaven lepšími znalostmi subjektivními, bude také lepším expertem.

Jak již bylo řečeno, expertní systémy jsou velmi rozšířenými nástroji umělé inteligence. Zatímco jejich první aplikace na počátku 80. let minulého století směřovaly hlavně do aplikační oblasti zdravotnictví, prosazují se nyní v průmyslu, bankovnictví a jiných technických i netechnických oborech [9].

Umělá inteligence a expertní systémy – shrnutí

Expertní systémy jsou určeny pro podporu rozhodování při řešení složitých situací a jsou nejvýraznějším příkladem využití metod umělé inteligence v praxi. Využívají znalostí, převzatých od experta a používají procedur, které je využívají při řešení konkrétního problému obdobně jako člověk-expert. Pro kvalitu rozhodování expertního systému mají zvláštní důležitost specifické až heuristické znalosti, které expert získává praxí.

Kontrolní otázky:

1. Jaký je účel expertních systémů?
2. Jaké znalosti experta mají pro kvalitu expertního systému největší význam?

Otázka k zamyšlení:

1. Vyjmenujte obor (obory), v nichž byste se označil jako expert.

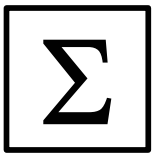
Úkoly k zamyšlení:

Pokuste se vysvětlit svůj myšlenkový postup při poslední konzultaci pracovního nebo osobního problému, při níž jste figuroval jako konzultant (poradce) - expert.

Korespondenční úkoly:

Napište pojednání o nejčastějším typu problémů, které řešíte v rámci vašich pracovních nebo jiných povinností a uveďte obory, jejichž znalost při řešení potřebujete.

Vyberte konkrétní problém a pokuste se popsat heslovitě Váš intelektuální postup jeho řešení.



Shrnutí obsahu kapitoly

Úvodní kapitola Vás seznámila s řadou důležitých základních pojmů, jejichž pochopení je nezbytné pro Vaše další studium. Nejdůležitější z nich jsou tyto:

- lidská (přirozená) inteligence a její projevy
- umělá inteligence, její zaměření a historický vývoj
- znalosti objektivní a subjektivní, znalosti heuristické
- strojové řešení problémů a hlediska jeho kvality
- expertní systém a jeho poslání

Doporučená literatura

[1] Feingenbaum,E.A.a kol.: The Rise of Expert Company, Times Book, London, 1988
[2] Rose,F.: Into the Hearth of the Mind, Harper & Row, NY, 1984
[3] Mařík,V. a kol.: Umělá inteligence I, ACADEMIA Praha, 1993
[4] Havel,I.M.: Robotika, SNTL Praha, 1983
[5] Minsky,M.: Computation. Finite and Infinite Machines, Prentice-Hall, Engelwood Cliffs,1967
[6] Rich,E., Knight,K.: Artificial Intelligence – Second Edition, McGraw Hill, Inc, NY, 1991
[7] Kotek,Z. a kol.: Metody rozpoznávání a umělá inteligence, In.: Kybernetika ve výzkumu a výuce, ČSVTS FE VŠSE, Plzeň, 1983
[8] Wiener,N.: Kybernetika, TKI, SNTL Praha, 1960
[9] Grosman,S. a kol.: Umělá inteligence a expertní systémy, KS Praha, 1990

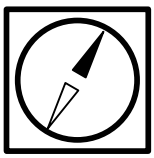
2 PRINCIPY EXPERTNÍCH SYSTÉMŮ

V této kapitole se dozvíte:

- Jaké třídy úloh jsou vhodné pro řešení pomocí expertních systémů
- Z jakých základních částí se expertní systémy skládají a jak fungují
- Principy a architektury diagnostických a plánovacích expertních systémů
- Které oblasti umělé inteligence tvoří teoretické zázemí expertních systémů

Po jejím prostudování byste měli být schopni:

- Dokázat vysvětlit význam znalostí pro práci expertních systémů
- Nakreslit a vysvětlit architekturu diagnostických a plánovacích expertních systémů
- Pojednat o základních oblastech umělé inteligence, které tvoří teoretické zázemí expertních systémů
- Získat ucelenou představu o účelu, konstrukci a funkci expertních systémů



Klíčová slova této kapitoly:
Znalosti objektivní a subjektivní, reprezentace znalostí, řídicí mechanizmus, strategie přímého a zpětného řetězení, neurčitost ve znalostech a datech, báze znalostí, báze dat, konceptualizace, ontologie, diagnostický a plánovací expertní systém.

Doba potřebná ke studiu: 5 hodin

Průvodce studiem

Studium a úspěšné zvládnutí látky v následující kapitole má stěžejní význam pro pochopení vlastních principů a funkcí činnosti expertních systémů. Postupujte pomalu, snažte se pochopit a rozumět jednotlivým částem probírané látky, využijte znalosti z kapitoly minulé. Pojmy vyznačené tučně jsou podstatné pro další studium. Využívejte vlastních zkušeností a hledejte vlastní příklady dokreslující látku.K prostudování této kapitoly si určete nejméně 5 hodin. K jejímu zvládnutí využívejte nejen textů této učebnice, ale také kapitol z doporučené literatury a konzultací s tutorem.

2.1 Myšlenka expertního systému

Příroda vytvořila (dosud ne zcela známými cestami evoluce) druh homo sapiens – člověka myslícího [1]. Ten v procesu vědeckotechnického rozvoje vytvořil fyzikální symbolové systémy – samočinné počítače. Pozorování a studium lidských myšlenkových (intelektuálních) procesů spolu se studiem symbolických procesů technicky realizovatelných vedlo ke vzniku a vývoji vědního oboru umělé inteligence (viz kap.1) a ke snahám o vytvoření „myslících“ počítačů.

Ve spojitosti s ideou expertního systému měly velký význam práce již zmíněného E. A. Feigenbauma, který počátkem 70. let minulého století řešil ve Stanfordu se svými spolupracovníky problém počítačové interpretace hmotových spektrogramů.

Ukázalo se, že snaha orientovat se na použití v té době již obvyklých, avšak obecných metod řešení problémů (např. systém GPS [2]), nevede zřejmě k uspokojivým výsledkům.

Východiskem řešení se stala skutečnost, že odborníci – experti ve svém oboru – spíše než metody všeobecné používají k (úspěšnému) řešení problémů metod specifických. Takové metody jsou založeny na specifických, subjektivních znalostech jejich oboru včetně řady hypotéz, které lze zdůvodnit buď obtížně nebo někdy je nelze zdůvodnit vůbec, avšak – jak z vlastní zkušeností poznali – často vedou k úspěšnému cíli. Feigenbaumův výzkumný tým se současně zabýval takovými metodami programování, které by umožnily začlenit do programu lidskou schopnost používat specifické znalosti. Výsledkem jejich úsilí se stal programový produkt DENDRAL [3] který se stal prototypem expertního systému.

Co jsou tedy expertní systémy? Uveďme několik vymezení, které se nejčastěji v odborné literatuře vyskytují:

Expertní systém je

- počítačový systém hledající řešení problému v rozsahu určitého souboru tvrzení nebo určitého seskupení znalostí, které byly formulovány experty pro danou specifickou aplikační oblast

- systém založený na reprezentaci poznatků expertů, které využívá při řešení zadaných problémů

- systém kooperujících programů pro řešení vymezené třídy úloh v určitých problémových oblastech, obvykle řešených experty

- počítačový systém vybavený znalostmi experta dané oblasti, v rozsahu kterých je schopen uskutečňovat rozhodnutí, rychlostí a kvalitou se expertovi vyrovnávajících.

Jak již bylo řečeno, expertní systémy jsou určeny pro podporu rozhodování při řešení složitých úloh. Základní **třídy problémů**, které jsou vhodné pro řešení expertními systémy, jsou:

interpretace	rozpoznávání situací z údajů, které je popisují
predikce	odvození očekávaných důsledků dané situace
diagnostika	určení stavu (poruchy) systému z pozorovatelných projevů jeho chování
konstruování	výběr a sestavení objektů do určitého funkčního celku při daných omezujících podmínkách
plánování	sestavení posloupnosti akcí za účelem dosažení požadovaného cíle

monitorování	sledování a porovnávání údajů odpovídajících určité situaci za účelem zjišťování a následného odstraňování odchylek od očekávané situace
ladění, opravování	výběr, sestavení a uskutečnění posloupnosti akcí, odstraňujících odchylky anebo chybové stavy
učení	diagnostika, ladění a upravování vědomostí studenta
řízení	interpretace, predikce, monitorování a úprava chování systému.

V průběhu vývoje expertních systémů a jejich aplikací došlo k celé řadě jejich velmi úspěšných a významných uplatnění, bohužel především v (západním) zahraničí. Zmínka o prvních úspěšných případech je uvedena v kap.1.2.

Myšlenka expertního systému - shrnutí

Myšlenka expertního systému je spojena s problémem řešení obtížných úkolů, vyžadujících od člověka řadu speciálních subjektivních znalostí. Počítačová realizace takového řešicího systému má několik definicí, všechny však zdůrazňují význam znalostí spojených s řešeným problémem a skutečnost, že expertní systémy mají omezenou působnost na velmi úzké a specializované problémy. Takové problémy lze rozčlenit do tříd, které jsou pro uplatnění expertních systémů typické.

Kontrolní otázky:

1. Pamatujete si alespoň tři definice zaměření expertních systémů?
2. Které třídy problémů pro expertní systémy jsou blízké vašemu profesnímu nebo osobnímu zaměření?

Otázka k zamyšlení:

1. Které z uvedených vymezení expertních systémů nejvíce odpovídá Vašemu názoru?

Úkoly k zamyšlení:

Pokuste se určit problém z uvedeného výčtu tříd problémů, v němž byste mohl působit jako expert.
Jakými cestami jste v jeho oblasti získával (-a) subjektivní znalosti?

2.2 Charakteristické vlastnosti a druhy expertních systémů

Hned v úvodu této kapitoly je třeba poznamenat, že cílem expertních systémů není přesné modelování mentálních pochodů v mozku člověka. Jejich úkolem je dosahovat při řešení složitých problémů obdobně kvalitních závěrů, kterých by dosáhl člověk – expert v daném oboru. Svým charakterem, strukturou i vlastnostmi se liší od konvenčních počítačových programů určených pro vědeckotechnické výpočty.

Expertní systémy jsou mj. typické tím, že lze nalézt celou řadu jejich společných charakteristických rysů. Věnujme se tedy rozboru základních charakteristických rysů expertních systémů podrobněji.

1. Není sporu o tom, že i konvenční programy využívají znalostí. Ty jsou však rozptýleny v jednotlivých instrukcích programu, které jsou aplikovány v přesně určeném pořadí (kap.3.1.). Naproti tomu u expertních systémů jsou znalosti experta vyjádřeny naprosto explicitně, jsou soustředěny v tzv. **bázi znalostí.** Předem je dána pouze strategie využívání takto uložených znalostí – **řídicí mechanizmus** (někdy také zvaný mechanizmus inferenční)**.** Striktní oddělení znalostí na straně jedné a strategie pro jejich využívání na straně druhé je pro architekturu expertních systémů naprosto typické. Poznamenejme již nyní, že tato separace je základem využitelnosti kvalitního, odladěného a prověřeného inferenčního mechanizmu (a dalších podpůrných komponent) i v případech aplikace na jiné báze znalostí (viz dále).

2. Báze znalostí obsahuje celou škálu znalostí experta (expertů) od obecných, objektivních až po soukromé, subjektivní. Řekli jsme již dříve, že velmi podstatné jsou heuristiky (tj. exaktně nedokázané nebo dokonce nedokázatelné znalosti), které expert získal dlouholetou praxí a o nichž pouze ví, že mu často pomáhají při řešení obdobných problémů, přičemž nalezení správného řešení přímo nezaručují. Znalosti experta nemají statický charakter, časem se vyvíjejí a rozvíjejí. Báze znalostí musí být tedy **modulární a otevřená** pro zahrnutí nových znalostí, případně eliminaci znalostí již neefektivních či redundantních.

3. Báze znalostí zahrnuje obecné znalosti daného oboru (jak uvidíme dále, lze ji považovat za obecný model problémové oblasti). Má charakter obecného rozhodovacího pravidla (či soustavy pravidel, jak uvidíme dále). Obdobně jako při simulaci chování soustavy popsané obecným matematickým modelem dosazujeme konkrétní (aktuální) vstupní data, řešení konkrétního případu znamená i v případě expertního systému „dosazení“ konkrétních dat o daném případu do obecně formulovaných pravidel báze znalostí. Množinu všech údajů vztahujících se k danému případu nazýváme **bází dat**.

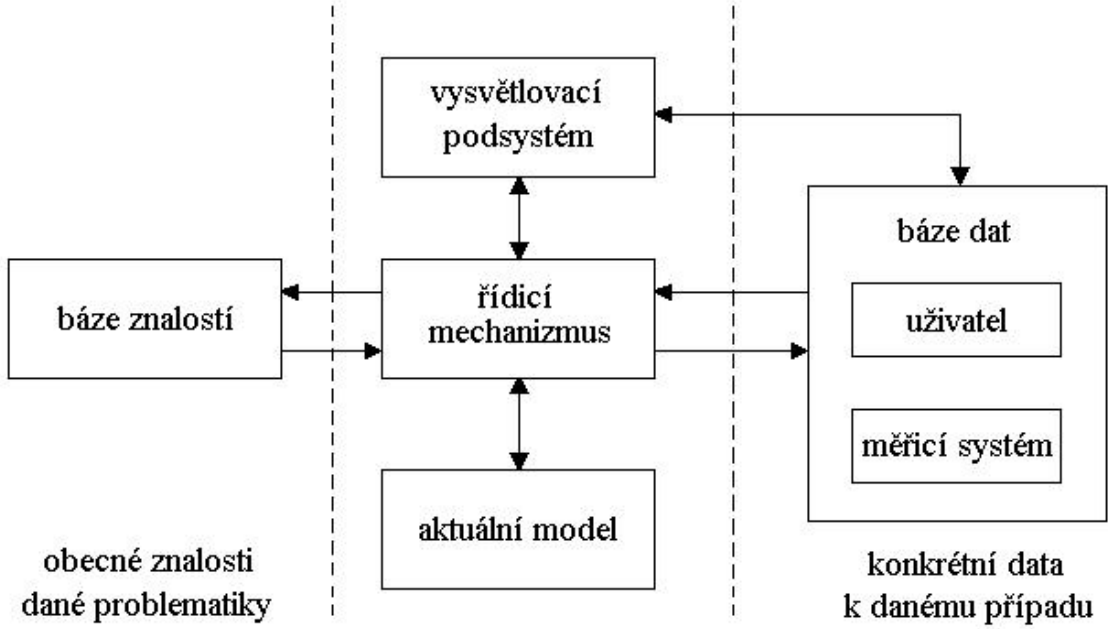
4. Způsob zpracování dat v expertním systému musí mít některé rysy způsobu uvažování experta. Expert především pracuje velmi často s nejistými znalostmi (heuristikami) i s nejistými daty (např. se subjektivním popisem potíží pacienta). Proto musí být expertní systém schopen pracovat s nejistotami**,** a to jak ve znalostech – tj. znalostí s přiřazenou mírou důvěry v její platnost (**nejistota v bázi znalostí**) tak i v aktuálních datech (tata typu „nevím“, „asi ano“, „spíše ne“ apod.), reprezentující **nejistotu v bázi dat**.

5. Stejně jako expert musí být schopen komentovat svá rozhodnutí a vysvětlit postup, jakým došel ke svým závěrům, tak i po expertním systému požadujeme, aby byl schopen podat informaci o způsobu a postupu svého uvažování vedoucího k závěru. Musí být schopen **zdůvodnit a vysvětlit** své závěry i dílčí kroky které k nim vedly. Vysvětlovací schopnost expertního systému je důležitá zvláště tehdy, pokud si uživatel ponechává prioritu při konečném rozhodnutí.

Je třeba poznamenat, že výčet těchto charakteristických vlastností nelze považovat za definici expertního systému jako takového. Expertní systémy dokonce nemusí splňovat striktně všechny uvedené znaky – některé z nich např. nepracuje s neurčitostí. Nelze však souhlasit s názorem, že každý konzultační program je expertním systémem.

Významným rysem expertních systémů je možnost jejich explicitního rozdělení podle charakteru řešených úloh do dvou skupin – na expertní systémy **diagnostické** a expertní systémy **plánovací** (vyloučena však není ani jejich hybridní varianta).

1. *Diagnostické expertní systémy* jsou určeny pro efektivní interpretaci dat s cílem určit, která z hypotéz (z předem stanovené konečné množiny hypotéz) nejlépe koresponduje s aktuálními daty týkajícími se konkrétního řešeného případu. Typickým použitím může být v tomto případě např. lékařský diagnostický systém pro určení druhu pacientovy choroby na základě vyhodnocení jeho subjektivních potíží. Typická architektura diagnostického expertního systému je uvedena na Obr.2.1.



Obr.2.1

Úlohou diagnostických expertních systémů je provádět efektivní interpretaci dat s cílem určit, která z hypotéz o chování zkoumané soustavy nejlépe koresponduje s reálnými daty. Řešení případu (problému) probíhá formou postupného ohodnocování a přehodnocování dílčích hypotéz v rámci pevně daného modelu řešeného problému, který je sestaven expertem.

Jádro takového systému tvoří **řídící (inferenční) mechanizmus**, který operacemi nad **bází znalostí** na základě aktuálních dat (dotazu) upřesňuje (aktualizuje) obecný model a vyvozuje **odpověď** (závěr).

Báze znalostí jako obecný model chování studované soustavy je tvořena expertními znalostmi, které jsou formalizovány vhodnou reprezentací. Kromě pravidlové reprezentace, která je preferována v této učebnici, existuje řada dalších možností.

Aktualizace modelu je provedena vstupem konkrétních dat k danému případu. Konkrétní data jsou reprezentována bází dat a mohou být získána jako jazykové hodnoty od uživatele (operátora), přímým měřením nebo kombinovaně.

Výsledkem činnosti diagnostického expertního systému je seznam ohodnocených závěrů - cílových hypotéz (diagnóz).

Uživatelsky významnou částí expertního systému je **vysvětlovací podsystém.** Ten poskytuje informace o konkrétním postupu, jímž bylo dosaženo závěru. Tak může uživatel sám posoudit kvalitu báze znalostí i inference a výsledek odvození případně dodatečně modifikovat.

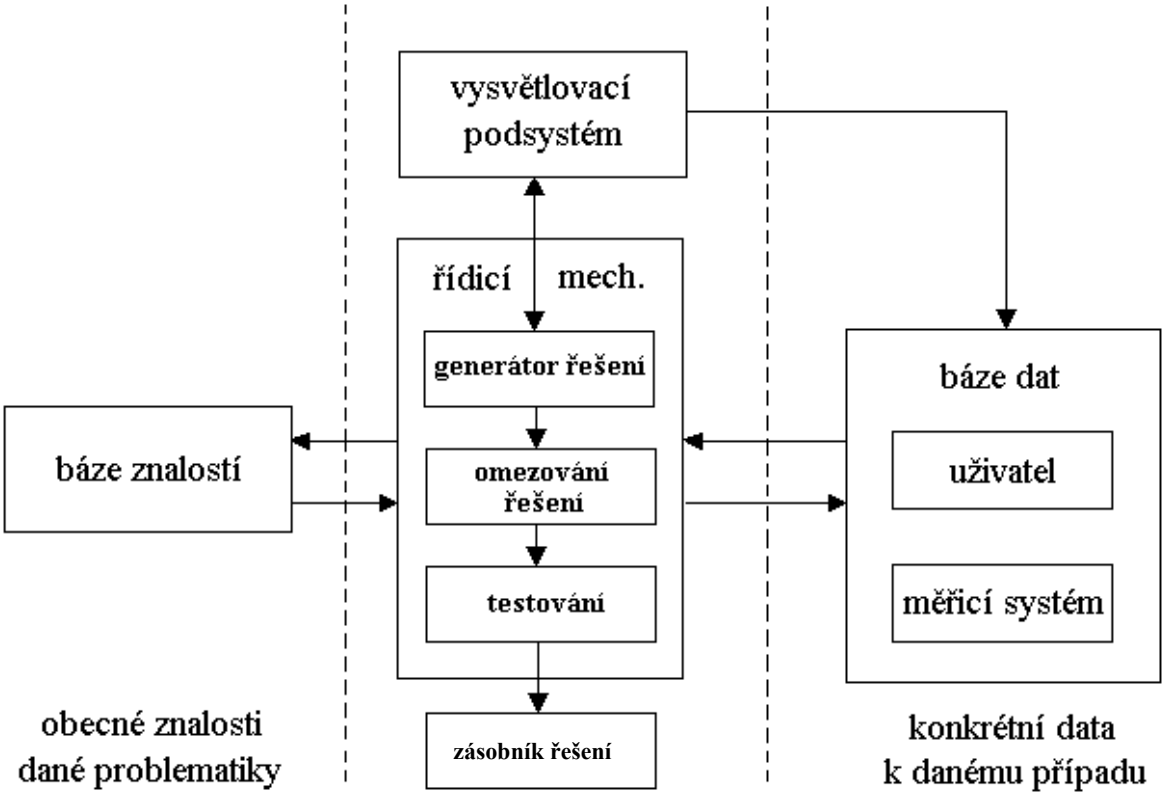
Aktuální model je velmi často vytvořen aktualizací hodnot jistot (vah, subjektivních pravděpodobností apod.) poznatků, zahrnutých v bázi znalostí. Na počátku mají jednotlivé poznatky z báze znalostí přiřazeny (expertem) apriorní hodnoty jistot a v průběhu procesu odvozování jsou tyto hodnoty měněny (to ostatně odpovídá i naší představě o modelu řešení. Na počátku, kdy o konkrétním případu není nic známo, odpovídá model pouze představám o podobných případech. Jakmile jsou získána konkrétní data o řešeném případu, jsou tyto představy modifikovány tak, aby konkrétnímu případu odpovídaly). Takto je aktuální model řešen např. u systémů MYCIN nebo PROSPECTOR, jimž jsou dále věnovány kap.4.1 a kap.4.2.

2. *Plánovací expertní systémy* jsou určeny pro řešení takových úloh, kdy je znám počáteční stav objektu a cíl řešení, přičemž systém má využitím konkrétních dat a zadání nalézt – pokud možno z nějakého hlediska optimální – posloupnost povolených kroků (operátorů), kterou lze cíle dosáhnout. Schéma struktury takového systému je uvedeno na Obr.2.2.

Základní částí plánovacího expertního systému je **generátor možných řešení**, který vytváří automaticky kombinace posloupností operátorů. Kombinatorická exploze řešení je omezována znalostmi experta i daty k reálnému případu. Navrhovaná řešení jsou posuzována z hlediska jejich přípustnosti a účelové funkce optimalizace řešení. Výsledkem činnosti plánovacího expertního systému je **seznam přípustných řešení**, z nichž každé je ohodnoceno stupněm kvality. Tento seznam je uložen v zásobníku vhodných řešení, jehož obsah se dynamicky mění.

Řídicí mechanizmus s využitím báze znalostí a báze dat

- ovlivňuje výběr přípustných operátorů
- omezuje generativní aktivitu generátoru řešení použitím znalostí z báze znalostí (např. apriorním zamítáním některých posloupností kroků)
- řídí testování shody generovaných řešení s daty z báze dat a utváří zásobník potenciálních přípustných řešení včetně ohodnocení jejich kvality.



Obr.2.2

Typickým a dnes již klasickým plánovacím expertním systémem je již zmiňovaný systém DENDRAL, vyvíjený na Stanfordské univerzitě v letech 1965 až 1969. Je určen k odvozování struktur chemických látek na základě histogramů rozložení hmotností získaných z hmotového spektrometru a ze spektrometru pro jadernou magnetickou rezonanci. Lze jej považovat za první expertní systém a dodnes se rutinně využívá [3].

3. *Hybridní expertní systémy* mají architekturu kombinovanou, využívají částečně principů diagnostických a částečně principů plánovacích. Např. u inteligentních výukových systémů se střídá diagnostika znalostí studenta s plánováním odpovídajícího dalšího výukového procesu. U systémů monitorovacích je cyklicky spouštěný diagnostický systém v okamžiku poruchy nahrazen systémem pro plánování zásahu k jejímu odstranění.

Velmi důležitým nástrojem v oblasti expertních systémů jsou tzv. **prázdné expertní systémy** (někdy nazývané „shell-systémy). Jejich idea vychází ze skutečnosti, že základní a zcela univerzální částí expertního systému je jeho inferenční mechanizmus. Ten může úspěšně operovat nad bázemi znalostí, které mají sice společnou architekturu, ale mohou být věcně (tj. problémově) orientovány do nejrůznějších oblastí. Prázdný expertní systém je vybaven řídicím mechanizmem a všemi ostatními komponenty, jeho báze znalostí je však „prázdná“. Naplněním báze znalostí konkrétními informacemi pak vzniká problémově orientovaný expertní systém, připravený k podpoře rozhodování v té problémové oblasti, na niž jsou znalosti jeho báze zaměřeny.

Prázdné expertní systémy přinesly v 80. letech minulého století velký rozmach jejich použití. Jelikož neobsahují znalosti (což je v podstatě know-how), jsou relativně levné a komerčně dostupné. Řešení jejich konkrétní problémové orientace, tj. naplnění báze znalostí a jejich odladění, spadá do poměrně nové samostatné součásti umělé inteligence nazývané **znalostní inženýrství**. Objevuje se tak nová profese znalostního inženýra jako odborníka na získávání, formalizaci, využívání a údržbu znalostí. Této problematice je věnována např. práce [4] nebo [5].

Charakteristické vlastnosti a druhy expertních systémů – shrnutí

Cílem expertního systému je dosahovat obdobně kvalitních závěrů při řešení složitých problémů, jako by stejný problém řešil člověk-expert v daném oboru. Hlavními částmi expertního systému je báze znalostí, řídicí (inferenční) mechanizmus, báze dat a vysvětlovací podsystém. Expertní systémy jsou schopny efektivně využívat eventuální nejistoty jak v bázi znalostí, tak v bázi dat. Principiálně je rozdělujeme na expertní systémy diagnostické a expertní systémy plánovací.

Kontrolní otázky:

1. Které jsou základní části každého expertního systému?
2. Jaký je rozdíl mezi bází znalostí a bází dat?
3. Jaké jsou základní třídy expertních systémů?

Otázka k zamyšlení:

1. Jakými nejistotami je zatížena Vaše odpověď ratolesti, která se Vás ptá na počasí „Jak je venku“?

Úkoly k zamyšlení:

Jaké informace byste požadoval, aby Vám je expertní systém poskytl při vysvětlování postupu stanovení závěru při řešení jemu položeného problému?

2.3 Teoretické zdroje expertních systémů

Expertní systémy jsou sice významným praktickým nástrojem umělé inteligence, nejsou však její samostatnou teoretickou partií. Jsou typickým aplikačním produktem jejích vybraných teoretických základů. Jsou to především problematiky

1. strojové reprezentace znalostí
2. strojového řešení úloh
3. zpracování neurčité informace.

Těmto třem oblastem jako teoretickým zdrojům expertních systémů budeme věnovat následující podkapitoly.

2.3.1 Strojová reprezentace znalostí

Reprezentace znalostí je spojena s problémem návrhu a konstrukce bází znalostí a je stěžejním problémem efektivity expertního systému. Požadavky na reprezentaci znalostí jsou následující:

- přirozený charakter reprezentace a její expresivita
- umožnění aplikace efektivních deduktivních prostředků
- rychlý přístup k položkám znalostí i dat.

Důležitým požadavkem je **modularita** báze znalostí. Do existující báze je třeba doplňovat znalosti nové a eliminovat znalosti již nepotřebné. Je to otázka údržby báze znalostí. Modulární struktura přináší výhody při změnách v bázi – jsou lokální a nepromítají se do ostatních částí báze.

Modulární uspořádání báze přináší jistou nevýhodu při jejím prohledávání. Jelikož znalosti modulárně uspořádané nezohledňují sémantiku báze, údaje týkající se jednoho objektu mohou být roztroušeny a při řešení musíme prohledat bázi celou.

Požadavek **sémantického sdružování znalostí** však nelze zcela opominout. Ten vyplývá především z požadavku na rychlost vybavování znalostí. Proto je nutno volit kompromis mezi modularitou báze znalostí a sémantickým sdružováním znalostí. Modularitu splňují např. produkční systémy využívající pro reprezentaci znalosti produkčních pravidel, požadavek sdružování zdůrazňují např. formalizmy sémantických sítí nebo rámců [6].

V souvislosti s reprezentací znalostí používáme často dělení reprezentace znalostí na znalosti reprezentované **deklarativně** a znalosti reprezentované **procedurálně.** Deklarativně reprezentované znalosti se nazývají **poznatky** které vyjadřují, co má být poznáno nebo dokázáno, např.:

Železo je kov

Tatáž znalost reprezentovaná procedurálně má podobu **podmíněného pravidla** typu JESTLIŽE-PAK (často IF-THEN):

JESTLIŽE (*X* je železo) PAK (*X* je kov).

Na uvedeném příkladu vidíme, že jednu a tutéž znalost můžeme někdy formalizovat jak deklarativně, tak i procedurálně. Často se také používají reprezentace kombinované.

Při řešení praktických úloh se vždy omezujeme pouze na část reálného světa. Některé objekty při řešení uvažujeme, jiné ignorujeme, protože se řešení netýkají. Soubor všech objektů tvoří **univerzum** dané úlohy, která je řešena pouze nad jeho objekty. Určení univerza definováním všech jeho objektů a vymezení funkcionálních vztahů či relací mezi nimi nazýváme **konceptualizací** [7], [8].

Ve filozofii je používán pojem **ontologie,** označující „zkoumání povahy existence“. Tento termín se používá v umělé inteligenci i pro explicitní popis konceptualizace. Pro expertní systém existují pouze pojmy, které ontologií popíšeme. Ontologie musí být reprezentována nezávisle na způsobu jejího budoucího použití. Bázi znalostí pak tvoří ontologie, doplněné znalostmi vztahujícími se ke způsobu řešení. Báze znalostí již není problémově nezávislá, je zaměřena na konkrétní typ úlohy a konkrétní způsob řešení.

Ontologii lze popsat jakýmkoliv, např. přirozeným jazykem. Vzhledem ke struktuře univerza může být přirozené použití jazyka predikátové logiky I. řádu. Problematika ontologií je zkoumána v rámci oboru znalostního inženýrství [5].

Jako příklad uveďme ontologii, týkající se benzínového motoru. Bude zahrnovat tyto pojmy:

motor
válec
píst
kliková hřídel
svíčky
ventily.

Vztahy budou vyjadřovat fakta typu

V každém válci je píst
Pokud je motor čtyřdobý, má sací a výfukové ventily
Každý válec má alespoň jeden sací a jeden výfukový ventil
Každý válec zážehového motoru má alespoň jednu svíčku
Každý pístový motor má klikovou hřídel

Taková ontologie je problémově nezávislá. Může sloužit více znalostním systémům, např. expertnímu systému pro diagnostiku poruch motoru, expertnímu systému pro konstrukci motoru, expertnímu systému pro podporu rozhodování při obchodování s motory atd.

2.3.2 Řídicí mechanizmy

Řídicí (odvozovací, vyvozovací, inferenční) mechanizmus zabezpečuje v expertním systému využívání znalostí. Při návrhu těchto mechanizmů se obvykle vychází z pojmů i výsledků **obecné teorie řešení úloh**, především z úlohy prohledávání stavového prostoru (prostoru řešení). Často jsou zahrnovány další, namnoze i ad-hoc principy a techniky řízení.

Ke konstrukci řídicích mechanizmů expertních systémů se využívá i technik, které z řešení problému prohledávání stavového prostoru bezprostředně nevycházejí. Jsou to např.:

*Technika agendy***.** V průběhu řešení se jako jeho vedlejší produkt vytváří hierarchický zásobník dalších úkolů, které by měly být postupně řešeny. Po vyřešení prvního dílčího úkolu se přistupuje k řešení úkolu dalšího (ležícího na vrcholu zásobníku). Princip využívání agendy se považuje za prostředek efektivního dynamického zaostřování pozornosti.

*Technika démonů***.** Metoda démonů vychází z představy fenoménu skřítků (démonů), kteří sledují průběh inferenčního procesu a zasáhnou do něj v případě předem přesně specifikované situace. Jsou představovány speciálními programy a v každém kroku inference musí být testováno, zda nenastala situace pro spuštění některého z nich.

*Technika nemonotónní inference***.** Inferenční proces probíhá v situaci, kdy je uvažována celá řada předpokladů a podmínek. V reálné situaci nemusí být některé z nich splněny a pak je nezbytné, aby řídicí mechanizmy byly vybaveny procedurami pro korektní úpravy aktuálního modelu po nesplnění původních předpokladů v průběhu řešení úlohy.

*Technika černé tabule***.** Znalostní báze je v tomto případě rozdělena na několik podbází, z nichž každá obsahuje znalosti z určité podoblasti řešené problematiky (znalostní zdroje). Komunikace mezi těmito dílčími bázemi je řízena speciální společně sdílenou datovou a řídicí strukturou, zvanou černá tabule. Systém využívající techniky černé tabule simuluje řešení problému panelovou diskuzí mezi odborníky několika souvisejících oborů.

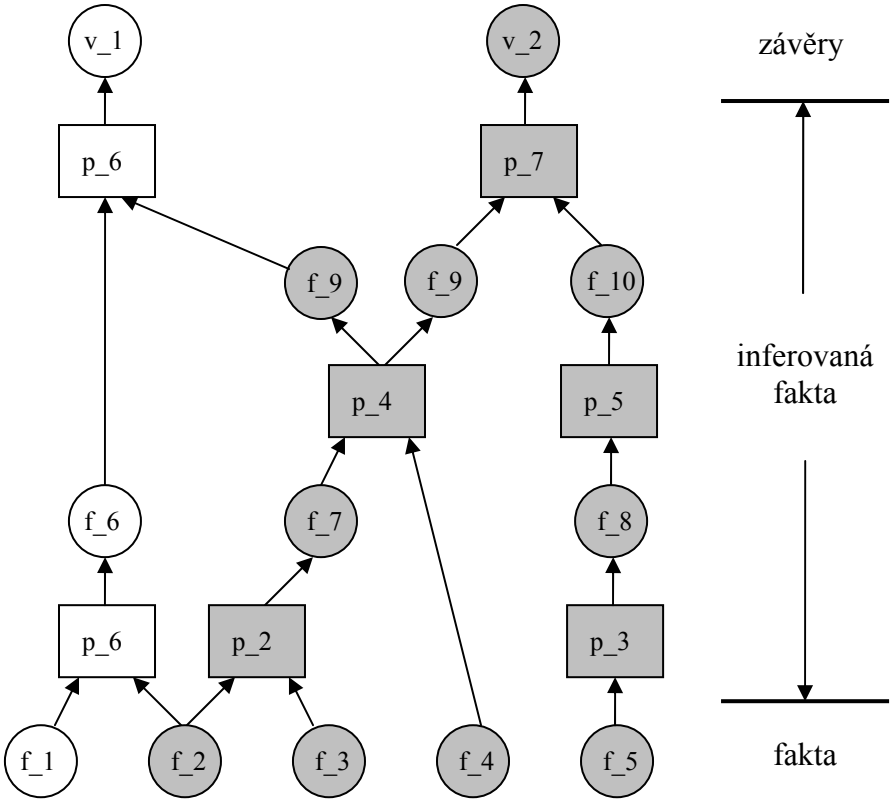
Technika taxonomie. Při použití této techniky je využito speciální formy odvozování tzv. děděním v taxonomické struktuře dílčích konceptů. Taxonomické struktury jsou určitou doplňkovou formou reprezentace znalostí a jsou využívány k zaostřování pozornosti v jednotlivých znalostních bázích. Úplné potvrzení některé položky v taxonomii v průběhu inference může mít za následek zamítnutí jiných položek na stejné taxonomické úrovni a jejich vyloučení z dalšího vyšetřování (ořezávání stavového prostoru).

V rámci pojednání o řídicích mechanizmech je nezbytné pojednat o základních strategií práce takových mechanizmů [9].

Inferenční mechanizmy pracují v jednotlivých krocích inference, které na sebe navazují. Posloupnost inferencí se nazývá **řetězec**.

Uvažujme situaci, kdy existuje skupina faktů, z nichž plynou určité závěry (hypotézy). Řekli jsme, že cílem expertního sytému je nalezení té z hypotéz, která nejlépe koresponduje s daty ke konkrétnímu případu. Toto vyhledání řídí inferenční mechanizmus. Při postupu směřujícím k dosažení svého cíle může použít dvě různé strategie, a to strategii **dopředného řetězení** nebo strategii **zpětného řetězení**.

V případě strategie dopředného (přímého) řetězení je proces inference řízen směrem od faktů k závěrům. Dopředné řetězení je určeno především k řešení problémů plánování, monitorování a řízení. Při něm se z faktů přítomnosti odvozují fakta budoucnosti. Jinými slovy, postupuje se od předcházejícího k následnému. Inference dopředného řetězení je proces řízený daty a postupuje „zdola nahoru“ – usuzování probíhá od faktů směrem k řešením. Příkladem dopředného řetězení je inference se sedmi pravidly, uvedená na Obr.2.3.



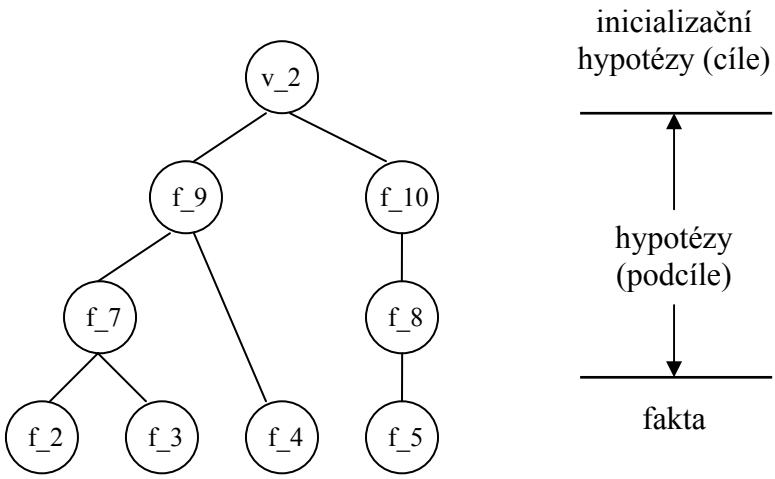
Obr.2.3

Uvažujme bázi znalostí, složenou z těchto pravidel:

pravidlo_1: *JESTLIŽE fakt_1 A fakt_1 PAK fakt_6*
pravidlo_2: *JESTLIŽE fakt_ 2 A fakt_3 PAK fakt_7*
pravidlo_3: *JESTLIŽE fakt_5 PAK fakt_8*
pravidlo_4: *JESTLIŽE fakt_4 A fakt_7 PAK fakt_9*
pravidlo_5: *JESTLIŽE fakt_8 PAK fakt_10*
pravidlo_6: *JESTLIŽE fakt_6 A fakt_7 PAK hypotéza_1*
pravidlo_7: *JESTLIŽE fakt_9 A fakt_10 PAK hypotéze_2*

Expertní systém je iniciován fakty *fakt_2, fakt_3, fakt_4* a *fakt_5*. Inference probíhá postupným plněním *pravidlo_2, pravidlo_3, pravidlo_4, pravidlo_5* a *pravidlo_7*. Systémem jsou vložena nová fakta *fakt_7, fakt_8, fakt_9* a *fakt_10*. Výsledkem inference je pak hypotéza *v_2*.

Strategie zpětného řetězení hledá cestu od hypotéz k faktům. Zpětné řetězení je také nazýváno inferencí „shora dolů“, usuzování probíhá od hypotéz směrem k faktům. Inference zpětným řetězením v expertním systému z předchozího příkladu je znázorněna na Obr.2.4.



Obr.2.4

Při inicializaci systému se předpokládá hypotéza *v_2*. Pro potvrzení správnosti výběru této hypotézy musí být splněna fakta *f_10* a *f_9*. Pro splnění faktu *f_9* musí být splněna fakta *f_7* a *f_4*. Podobně se postupuje pro další fakta.

2.3.3 Zpracování neurčité informace

Jak jsme již zmínili výše, pro expertní systémy je zcela typická práce s neurčitou, nepřesnou nebo chybějící informací. Znalosti jsou heuristické, statisticky nepodložené, vágní, data jsou zašuměná nebo zkreslená vlivem použitých metod či chyb přístrojů.

Práce s neurčitostí není cizí ani klasické matematice. Jejím nástrojem pro řešení problémů inference v podmínkách neurčitosti **je matematická pravděpodobnost.** Fundamentální rovnice definující pravděpodobnost vzniku jevu E při provedení N pokusů je dána vztahem

$$p(E) = \frac{E}{N}$$

V klasické pravděpodobnosti platí tři základní axiomy. První lze vyjádřit vztahem

$$0 \le p(E) \le 1$$

Druhý axiom je představován vztahem

$$\sum_i p(E_i) = 1$$

Třetí axiom uvádí vztah pro výslednou pravděpodobnost nezávislých jevů E_1 a E_2 ve tvaru

$$p(E_1 \cup E_2) = p(E_1) + p(E_2)$$

Podmíněná pravděpodobnost je nástroj pro popis jevů, které se vzájemně ovlivňují. Je definována jako pravděpodobnost vzniku jevu *A* za podmínky, že nastane jev *B*

$$p(A \mid B) = \frac{p(A \cap B)}{p(B)}$$

kde $p(B)$ je apriorní pravděpodobnost (nepodmíněná) vzniku jevu *B*.

S podmíněnou pravděpodobností pracuje tzv. **Bayesův vztah pro dokazování hypotéz** [8]**.** Tento vztah je klasickou obdobou expertního systému pro ohodnocování cílových hypotéz. Předpokládejme, že vznik jevu *E* (evidence) může být vysvětlen některou z předem stanovených n- hypotéz H_i , i = 1,2,…, n. Pravděpodobnost platnosti *i*-té hypotézy pro vysvětlení evidence *E* je dán Bayesovým vztahem

$$\begin{aligned} p(H_i|E) &= \frac{p(E \cap H_i)}{\sum_j p(E \cap H_j)} \\ &= \frac{p(E|H_i)p(H_i)}{\sum_j p(E|H_j)p(H_j)} \\ &= \frac{p(E|H_i)p(H_i)}{p(E)} \end{aligned} \qquad (2.1)$$

Bayesův vztah platí v klasické teorii pravděpodobnosti. Především platí zákon, uvádějící vztah mezi pravděpodobností jevu $P(A)$ a pravděpodobností jeho negace $P(\text{neg}A)$.

$P(\text{neg}A) = 1 - P(A)$.

Pro účely použití Bayesova vztahu v oblasti umělé intelligence dochází k jeho modifikacím, jak uvidíme v kap.4.2.

Nejistota bývá v expertních systémech vyjadřována různými způsoby:

- především to mohou být různé váhy, míry, stupně důvěry, faktory jistoty či jinak nazývané a formulované **subjektivní pravděpodobnosti**. Tyto numerické parametry jsou přiřazovány k elementům báze znalostí, např. k jednotlivým tvrzením, pravidlům či rámcům. Číselný interval těchto parametrů bývá obvykle $\langle 0, 1 \rangle$ nebo $\langle -1, 1 \rangle$ - viz kap.4.

- jsou-li k reprezentaci znalostí použity popisy s využitím **přirozeného jazyka** (formou jazykových podmíněných pravidel), pak je jejich neurčitost dána stupněm vágnosti použitých jazykových termů („velmi vysoký“, „ne zcela přesný“, „přibližně nulový“ apod.). K formalizaci takových vágních slovních pojmů používáme prostředků fuzzy množinové matematiky, inferenční mechanizmy pak využívají principů fuzzy jazykové logiky – viz kap.5.

Teoretické zdroje expertních systémů - shrnutí

Teoretickými zdroji expertních systémů jsou vybrané oblasti zájmu umělé inteligence, jmenovitě věda o strojové reprezentaci znalostí, strojovém řešení úloh a zpracování neurčité informace. Znalosti bývají uloženy v bázi znalostí nejčastěji formou produkčních pravidel. Mohou být formulovány deklarativně nebo procedurálně. V souvislosti s problematikou reprezentace znalostí hovoříme o konceptualizaci báze znalostí a její ontologii. Řídicí mechanizmy, operující nad bází znalostí, mohou pracovat s využitím několika principů. Pracují v režimu přímého nebo zpětného způsobu řetězení kroků inference. Formalizace neurčitosti se provádí nejčastěji využitím subjektivních ohodnocovacích parametrů, jako jsou stupně důvěry, váhy apod. Způsoby jejich zpracování se odchylují od axiomů klasické pravděpodobnosti.

Kontrolní otázky:

1. Jaké jsou základní zdroje teoretického zázemí expertních systémů?
2. V čem spočívá rozdíl mezi procedurální a deklarativní reprezentací znalostí?
3. Co je ontologie?
4. Čím se odlišují strategie řídicího mechanizmu při použití metod přímého a zpětného řetězení?

Otázka k zamyšlení:

1. Používá člověk při řešení problémů techniky nemonotónní inference?
2. Jsou strategie přímého a zpětného řetězení zaměnitelné?

Úkoly k zamyšlení:

Formulujte alespoň tři podmíněná pravidla, jimiž vyjádříte znalosti používané v oboru vašeho hobby.

Korespondenční úkoly:

Vyberte konkrétní problém ze svého okolí a popište systematicky Váš postup při jeho jeho řešení. Uveďte oblasti znalostí, které jste při jeho řešení museli použít a soustřeďte se hlavně na znalosti subjektivní

Shrnutí obsahu kapitoly

Druhá kapitola Vám přinesla velmi důležitou představu jak o poslání, tak o strukturách a funkci jednotlivých částí expertních systémů. Nejdůležitější pojmy jsou

- typy znalosti a jejich dělení na objektivní a subjektivní,
- problematika počítačové reprezentace znalostí
- procedura usuzování expertního systému
- strategie přímého a zpětného řetězení v procesu inference,
- neurčitosti ve znalostech a datech
- báze znalostí, řídicí mechanizmus a báze dat
- konceptualizace a ontologie
- diagnostický a plánovací expertní systém.

Doporučená literatura

[1] Popper,M., Kelemen,J.: Expertné systémy, ALFA Bratislava, 1988
[2] Newel,A., Simon,H.A.: GPS, a programm that simulates human thought, Prentice Hall, NJ, 1963
[3] Feingenbaum,E.A., a kol.: On generality and problem solving: a case study using the DENDRAL programm, Machine Inteligence, Vol. 6, Edinburgh University Press, 1971
[4] Mařík,V. a kol.: Umělá inteligence II, ACADEMIA Praha, 1997
[5] Vlček,J.: Znalostní inženýrství, ČVUT Praha, Edice Neural Network World, Praha, 2003
[6] Mařík,V. a kol.: Umělá inteligence I, ACADEMIA Praha, 1993
[7] Klir,G.J.: Architecture for Systems Problem Solving, Plenum Press, NY, 1985
[8] Kotek,Z.a kol.: Kybernetika, SNTL Praha, 1987
[9] Provazník,I., Kozumplík,J.: Expertní systémy, VUT Brno, FEI, 1999.

3 JAZYKY PRO EXPERTNÍ SYSTÉMY

V této kapitole se dozvíte:

- Zásady procedurálního a deklarativního programování
- Principy logického programování postupů řešení v rámci umělé inteligence
- Základy tvorby programů v jazyku PROLOG
- Přístupy programování v jazyku LISP

Po jejím prostudování byste měli být schopni:

- Vysvětlit přístupy deklarativního a logického programování nástrojů umělé inteligence
- Vypracovat jádro jednoduchého programu v jazyce PROLOG
- Pojednat o principech programovacího jazyka LISP

Klíčová slova této kapitoly:
Procedurální programování, deklarativní programování, logické programování, PROLOG, LISP

Doba potřebná ke studiu: 6 hodin

Průvodce studiem

Následující kapitola Vás seznámí s principy tvorby programového vybavení systémů umělé inteligence. Představí Vám nový přístup deklarativního programování, uvede programování logické a na příkladu konstrukce a použití programovacího jazyka umělé inteligence PROLOG Vás seznámí s jeho principy. Uvedeme příklady praktických prologovských programů určených pro řešení vědeckotechnického problému i jednoduchého expertního systému. Pro srovnání uvedeme i principy obdobného programovacího jazyka LISP.
Úkolem kapitoly není naučit Vás dokonale programovat v PROlOGu, cílem je objasnit principy programů pro zpracovaní znalostí a připravit Vás pro eventuální další studium.
Předpokládaná doba zvládnutí kapitoly je asi 6 hodin. Využijte opět doporučené literatury a konzultací s tutorem.

3.1 Programovací jazyky umělé inteligence

Jednou z podstatných změn, které přinesla umělá inteligence ve využití potenciálu počítačů, byla změna v představách o tradičním způsobu programování [1] - viz kap.2.2. Východiskem byla teze Minského [2], který jmenoval a označil tři přístupy programování s principy

- proveď ihned
- proveď, když můžeš
- proveď něco smysluplného.

První případ odpovídá způsobu klasického **procedurálního programování**. Program je tvořen posloupností příkazů, z nichž některé mohou tvořit cykly. Tento způsob vyžaduje od programátora přesnou představu všech situací, do nichž se zpracování dat může dostat a také způsobů, jak se v těchto případech bude chovat program. Jde o případ zpracování dat řízené programem. V každém kroku je určeno, co se má vykonat v kroku následujícím. Princip řízení programu lze slovně formulovat takto:

Jestliže další krok postupu není explicitně definován programovou konstrukcí, proveď následují instrukci.

Příklad procedurálního programování ukážeme na programu úlohy výběru vysokých jedinců ze seznamu *Z*, přičemž za vysokého budeme považovat toho, kdo je vyšší než 175cm:

```
proveď
        J : = 0
        pro I od I = 1 po I = MAX
        proveď
                jestliže VÝŠKA (I) ≥ 175cm
                proveď
                        J : = J + 1
                        VYSOKÝ (J) : = JMÉNO (I)
                        konec
        konec
konec
```

Základní představa druhého přístupu, tzv. **deklarativního programování** (někdy nazývaného programováním situačním) je jiná. Neorientuje se na proceduru programu, nýbrž na zákonitosti (pravidla) řešení problémů, které odděluje od vlastních řešicích procesů. Používá logické formalizmy, jejichž seskupení tvoří deklarativní program. Nutnou součástí deklarativních programů je interpretátor použitých symbolicky vyjádřených zákonitostí. Ten je v procesu řešení interpretuje spolu s důsledky, které z nich plynou. Automaticky inicializuje a koordinuje procedury které realizují procesy, potřebné k řešení problému.

Jednoduchý deklarativní program, ekvivalentní předešlému programu procedurálnímu, používá fenomén seznamu *Z* a může mít např. takový tvar:

PRO-KAŽDÉ ($x \in Z$) **JESTLIŽE** *VÝŠKA* ($x \geq 175$cm) **PAK** *VYSOKÝ* (x)

Způsob deklarativního programování zbavuje programátora nutnosti předem předvídat tok řízení při provádění programu, tj. vytváření infrastruktury programu, která deterministicky předdefinovaným způsobem zabezpečuje dosahování zadaných cílů.

Programátor vhodnými pravidly předepíše akce (operace nebo skupiny operací), odpovídající jednotlivým situacím, do nichž se řešení problému může dostat. Přitom nemusí předvídat, kdy takové situace nastanou. Vyspělejší prostředky situačních programů nevyžadují po programátorovi dokonce ani předvídat jednotlivé situace řešení problému. Stačí, aby dokázal formulovat úlohu natolik jednoznačně, aby použitému programovému systému bylo zřejmé, **co** se má řešit bez nutnosti formulování **jak** dospět k výsledku řešení. Systém sám vyvodí vzájemné vazby mezi situacemi a akcemi, které je při jejich vzniku účelné vykonat.

Při situačním řešení problému (s použitím deklarativního programování) se nevyžaduje, aby byly předem deterministicky určeny jednotlivé operace a pořadí jejich provádění. Operace se provádějí tehdy, kdy systém dospěje k situacím, které jejich vykonání vyžadují. Zjednodušeně lze opět princip situačního řešení problémů vyjádřit např. takto:

Jestliže mezi aktuálním a požadovaným stavem řešení vznikl rozdíl, pak jej odstraň vhodnou akcí (operací).

Pokud se týká expertních systémů, pak prostředky, umožňující realizaci systémů situačního řešení problémů, představují jeden z hlavních předpokladů jejich vzniku. Můžeme říci, že expertní systémy jsou svojí podstatou ztělesněním principů deklarativního programování.

Poznamenejme, že třetí postup programování podle Minského je použití programů, které se dovedou učit a usuzovat na základě analogií. Hovoříme o principu **evolučního programování**. Jeho princip spočívá ve schopnosti naučit se rozpoznávat, které předchozí „zkušenosti" (uchované v paměti) jsou na základě nejvhodnější analogie použitelné při řešení daného problému. Slovně lze princip evolučního programování vyjádřit takto:

Jestliže je zadaný problém analogický s problémem, v minulosti úspěšně vyřešený metodou M, pak se pokus tuto metodu použít i při řešení daného problému.

Bližší informace o této problematice může najít čtenář např. v [1].

Programovací jazyky umělé inteligence - shrnutí

Umělá inteligence přinesla zásadní změnu v nárocích na programování, která spočívá v nutnosti přechodu od zpracování dat ke zpracování informací (znalostí). Tento požadavek se odrazil v odklonu od klasického programování procedurálního a přechodu na programování deklarativní. Jeho základem je oddělení zákonitostí řešení problému od vlastních řešicích procesů. Programový systém formalizuje otázku co se má řešit bez nutnosti explicitního a detailního předpisu jak dospět k výsledku řešení.

Kontrolní otázky:

1. Jste schopen vlastními slovy vysvětlit rozdíl mezi deklarativním a procedurálním způsobem programování?
2. Jaký je význam situačního programování v prostředcích umělé inteligence?

Otázka k zamyšlení:

1. Používáte ve své práci metody procedurálního či dokonce deklarativního programování? V jakém jazyce?

Úkoly k zamyšlení:

Napište příklad procedurálního a deklarativního zápisu programu jednoduché úlohy, kterou sami navrhnete.

3.2 Logické programování

I když lze jakýkoliv počítačový program vytvořit v libovolném (klasickém) programovacím jazyku, je použití těchto jazyků pro realizaci nástrojů umělé inteligence neefektivní. Charakter programů nástrojů umělé inteligence totiž většinou vyžaduje – na rozdíl od klasických výpočtů – efektivní práci s daty v symbolickém tvaru a s bohatou vnitřní strukturou (seznamy, řetězce, stromy), mechanizmus navrácení při prohledávání prostorů řešení, asociativní paměť a mechanizmus pro automatickou dedukci. Proto vznikly speciální jazyky pro softwarovou realizaci nástrojů umělé inteligence - počátkem 60. let minulého století LISP, později např. POP-2, PLANNER a konečně velmi rozšířený PROLOG.

Programovací jazyk PROLOG, který vychází z formulí jazyka predikátové logiky 1. řádu, vznikl koncem 70. let minulého století v USA původně jako prostředek pro analýzu přirozeného jazyka. V průběhu doby se z něj stal univerzální programovací prostředek pro softwarovou realizaci celé řady úloh umělé inteligence (úlohy hledání řešení problémů, expertní systémy, zpracování přirozeného jazyka, rozpoznávání obrazu, aplikace v učicích se systémech a aplikace v úlohách robotiky [7]).

Programovací jazyk PROLOG [3], [4], [5], [6], je reprezentantem přístupu **logického programování**, ustupující od přístupu klasického programování procedurálního. Využívá původní deklarativní přístup programování, kdy těžiště práce programátora je soustředěno pouze do nalezení vhodného popisu řešené úlohy a vztahů v ní.

Hlavní rysy programovacího jazyka PROLOG jako reprezentanta přístupu logického programování lze vyjádřit těmito jeho vlastnostmi:

- uživatel jazyka PROLOG pouze specifikuje **čeho** má být při řešení úlohy dosaženo (znak deklarativního přístupu) aniž by musel rozepisovat postup, **jak** toho má být dosaženo (znak procedurálního přístupu)

- není tudíž nutno formulovat úlohu ve formě algoritmů s využíváním řídicích struktur klasických programů

- jednoduchost syntaxe jazyka PROLOG spolu s možností formulovat relace v přirozeném jazyce velmi zefektivňují práci programátora i uživatele

- schopnost PROLOGu modifikovat program během jeho vykonávání přináší výhodnou vlastnost adaptability.

Všechny tyto uvedené vlastnosti přinášejí možnost reprezentovat řešenou úlohu formou **symbolického popisu**, který tvoří základ metod umělé inteligence [8].

Základem logického programování jsou termy a výrazy, které jsou reprezentovány

fakty,
dotazy,
pravidly.

Fakta jsou nejjednodušší výrazy, vyjadřující vztah (relaci) mezi objekty. Např. výraz

otec(petr, pavel)

znamená, že

Petr je otec Pavla

neboli relace *otec* vyjadřuje vztah mezi objekty *Petr* a *Pavel*. Místo pojmu relace používáme pojem **predikát** a jména objektů nazýváme **atomy**. Důležitá je konvence používání malých a velkých písmen – malá vyjadřují jména predikátů a jejich argumentů a velká reprezentují proměnné.

Nejjednodušší program v jazyce PROLOG pak představuje konečná množina takových faktů. Příklad je následující:

o*tec(petr, pavel)*	*muž(pavel)*
otec(jan, ivan)	*muž(ivan)*
otec(jan, marie)	*žena(marie)*
otec(jan, eva)	*žena(eva)*

Jak vidíme z tohoto příkladu, predikát může mít i charakter unární (obsahuje pouze jeden argument) nebo dokonce nemusí mít argument žádný.

Dotaz je další formou logického výrazu. Jeho význam spočívá v podání odpovědi, zda určitá forma predikátu vyjadřuje vztah mezi objekty nebo ne. Tak např. na dotaz

?-otec(petr, pavel)

v našem programu bude následovat kladná odpověď

yes.

Dotaz se z hlediska syntaxe jazyka PROLOG liší od faktu předřazením *?-* a označuje se anglickým **goal** (cíl). Rozhodnutí o kladné nebo záporné odpovědi přitom závisí na tom, zda v programu existuje nebo neexistuje totožnost (**identita**) položeného dotazu a některého faktu.

Kromě jednoznačně specifikovaných objektů lze v jazyce PROLOG použít i **proměnné** pro objekty nespecifikované. Použijeme-li v dotazu

?-otec(petr, X)

dostaneme odpověď

X=pavel.

Proměnná v dotazu hledá takové identity, které splňují podmínku identity.

V jazyku PROLOG můžeme použít i **proměnné anonymní**, reprezentované symbolem _ , na jejíž hodnotě vůbec nezáleží. Význam anonymní proměnné ukážeme na dotazu

?-otec(petr, _)

kterým nechceme zjistit konkrétní potomky *Petra* ale zjišťujeme, zda *Petr* je vůbec otcem.

Dosud probrané základy programování v PROLOGu můžeme nyní shrnout do jednoduchých datových struktur, zvaných **termy**. Jsou tvořeny

konstantami (*atomy*),
proměnnými,
složenými termy.

Složené termy se v syntaxi PROLOGu skládají z **funktoru** *f* a posloupnosti **argumentů** t_i, které jsou opět reprezentovány termy, tedy

$f(t_1, t_2, \ldots, t_n)$.

Uveďme příklady takových termů:

muž(pavel)
kniha(autor(alois, jirásek), název(staré_pověsti_české))
strom(a, strom(b, strom(c, d))).

Důležitá je skutečnost, že proměnné jsou obecně substituovány termy. Na dotaz

?-kniha(autor(X, jirásek), Y)

dostaneme tedy odpovědi

X=alois
Y=název(staré_pověsti_české).

Kromě samostatných dotazů (cílů) můžeme vytvářet i **konjunkce dotazů** (cílů) jako jejich posloupnost oddělených čárkou. Podmínkou kladné odpovědi je pak splnění všech dotazů. Např. v konjunkci dotazů

?-(otec(jan, X), muž(X)

vede k odpovědi

X=ivan

což v našem příkladu znamená, že druhý dotaz v pořadí omezuje množinu substitucí dotazu prvního.

Pravidla jsou typem výrazů, které umožňují definovat nové vztahy mezi termy na základě termů které již existují. Pravidlo v PROLOGu má následující schéma:

$A \leftarrow B_1, B_2, \ldots, B_n$

Přičemž A je **hlava pravidla** a posloupnost B_i tvoří **tělo pravidla**. Důležité je, že pravidlo můžeme interpretovat větou přirozeného jazyka

" A platí, jestliže platí $B_1, B_2, \ldots, B_n$ ".

Způsob, jakým pravidla definují nové vztahy, ukážeme na následujícím příkladu:

o*tec(petr, pavel)*	*muž(pavel)*
otec(jan, ivan)	*muž(ivan)*
otec(jan, marie)	*žena(marie)*
otec(jan, eva)	*žena(eva)*

syn (X, Y) ← otec(X, Y), muž(X)
dcera (X, Y) ← otec(Y, X), žena(X)
praotec (X, Z) ← otec(X, Y), otec(Y, Z)

Dotaz

?-syn(X, jan)

(*kdo je synem Jana?*) položený programu, který obsahuje dané pravidlo, vede k jeho převodu na plnění konjunkce dotazů

otec(jan, X), muž(X)

Odpověď na takový dotaz (dokázání platnosti dotazu resp. hlavy pravidla) hledá PROLOG s využitím principu logické dedukce, který se nazývá **modus ponens**. Ten spočívá v tvrzení, že

z platnosti B a existence A ← B vyplývá A.

S tímto pravidlem se setkáme dále i v kap.5.7. Pro náš příklad tedy z existence faktů

muž(ivan)

a

otec(jan, ivan)

vyplývá z principu modus ponens platnost dotazu (odpověď)

X=ivan.

Uvědomíme si, že takovou dedukci můžeme vyjádřit větou přirozeného jazyka:

Jestliže X je otcem Y a X je muž, pak synem Y je X.

Taková věta představuje myšlenkový postup (inteligentního) člověka, hledajícího odpověď na daný dotaz a používajícího znalostí, které má jako fakta uloženy ve své paměti. Pak lze tedy uvedenou strojovou dedukci považovat za nástroj (postup) umělé inteligence.

Predikát může mít někdy více realizací, např.:

rodič(X, Y) ← otec(X, Y)
rodič(X, Y) ← matka(X, Y).

fakta, dotazy a pravidla se také nazývají **Hornovy klauzule** nebo zkráceně **klauzule** a samotný logický program jazyka PROLOG je jejich konečná množina.

Logické programování - shrnutí

Charakter programů umělé inteligence vyžaduje efektivní práci s daty v symbolickém tvaru. Proto vznikly speciální programovací jazyky, např. jazyk LISP nebo PROLOG. Kapitola uvádí principy jazyka PROLOG jako reprezentanta jazyků logického programování, který při tvorbě programů důsledně využívá deklarativní přístupy. Jsou uvedeny hlavní rysy PROLOGu a definovány jeho základní pojmy - fakta, dotazy a pravidla. Princip jejich využití při konstrukci programu je ilustrován jednoduchým příkladem.

Kontrolní otázky:

1. Které jazyky pro realizaci nástrojů umělé inteligence znáte?
2. Jaké jsou hlavní rysy jazyka PROLOG jako jazyka pro logické programování?
3. Dokážete formulovat princip logické dedukce nazvaný modus ponens?

Otázka k zamyšlení:

1. Vyžadují postupy lidského uvažování i jiné logické principy než je modus ponens?

Úkoly k zamyšlení:

Sestavte schéma logické dedukce při řešení některého Vašeho problému s použitím principu modus ponens.

3.3 Základy programování v PROLOGu

Databáze programu

Podle závěru minulé kapitoly se program v jazyku PROLOG skládá z klauzulí, které jsou reprezentovány fakty a pravidly, a dále z dotazů, které umožňují jejich využívání. Reprezentace faktu je pak následující:

funktor(arg1, arg2, ..., argN)

Reprezentace pravidla má tvar

hlava :– tělo

kde **hlava** popisuje vztah (relaci), který má být dokázán, **tělo** pak popisuje konjunkci subcílů, které musí být postupně (jeden po druhém) splněny. Symbol *:–* je přitom ekvivalentní se symbolem ← z minulé kapitoly. Vlastní konjunkce subcílů je realizována pomocí oddělovací čárky.

Zvláštní formou programu v jazyku PROLOG je tzv. **seznam.** Je to uspořádaná množina programových prvků a může mít libovolnou délku. Prvky mohou být atomy, složené termy nebo opět seznamy. Seznam je buď **prázdný seznam,** který nemá žádný prvek, nebo je to **struktura**, která má hlavu a tělo, kde hlava je první prvek seznamu a tělo je pak jeho zbytek.
Syntakticky je seznam vyjádřen buď výčtem všech jeho prvků v hranatých závorkách

[první, druhý, třetí]

nebo pomocí rozdělení „/“ na hlavu a tělo

[první | X],

kde *první* reprezentuje hlavu a *X* reprezentuje tělo ve formě seznamu

[druhý, třetí].

Práci se seznamy můžeme ukázat na příkladu, který testuje výskyt určitého prvku v seznamu, přičemž současně uvedeme i **princip rekurze** jako jednu ze základních metod jazyka PROLOG:

```
/* Mezní podmínka – Prvek je v hlavě seznamu */
        prvek(Prvek, [Prvek |_ ].
/* Druhá podmínka – rekurze */
        prvek (Prvek, [_ |Seznam] ) :–
                prvek(Prvek, Seznam) .
```

První klauzule testuje, zda je první prvek seznamu (hlava) totožný s ověřovaným prvkem (Prvek). V opačném případě dochází v druhé klauzuli k odebrání prvého prvku seznamu a rekurzivnímu ověřování predikátu **prvek**, tentokráte se zkráceným seznamem. Postupně tak dochází k redukci původního seznamu až do okamžiku, kdy

bude první prvek aktuálního seznamu totožný s prvkem ověřovaným, nebo vstupní seznam bude prázdný a systém ohlásí neúspěšnost dokazování. Odpovědí na dotaz

?- prvek(druhý, [první, druhý, třetí]

tedy bude

yes.

Unifikace a vázání proměnných

Výběr informace z programu (databáze) se provádí vyhledáním prvého výskytu klauzule, která je **unifikovatelná** s cílem.

- pokud je touto klauzulí *fakt*, pak je cíl splněn.
- pokud je touto klauzulí *pravidlo*, pak je splnění požadovaného cíle podmíněno splněním všech subcílů tvořících tělo daného pravidla.

Pravidla unifikace jsou následující:

- dvě konstanty jsou unifikovatelné pouze když jsou shodné

- nevázaná proměnná (nezastupující žádný objekt) je unifikovatelná s libovolným objektem. Tím se proměnná *váže* na tento objekt a na tento objekt jsou také vázány všechny její výskyty v dané klauzuli (**instalování proměnné**).

- dvě nevázané proměnné se unifikováním stávají **sdruženými**, tedy v případě instalování některé z nich dojde k instalování i zbývajících

- struktury jsou unifikovatelné v případě totožného funktoru a stejného počtu argumentů, přičemž každý z nich musí být opět unifikovatelný.

Uveďme si příklad: unifikace dvou objektů

kniha(autor(X, Y), Z) *kniha(autor(alois, jirásek), W)*

při splnění totožnosti jmen funktorů a počtu jeho argumentů vede k výsledku

X=alois	instalování proměnné X
Y=jirásek	instalování proměnné Y
Z=W	sdružení proměnných Z a W

Princip navrácení

Při prohledávání databáze postupuje systém **od shora dolů a zleva doprava**, vždy pro každý nový cíl (subcíl) od začátku. V případě neúspěšného pokusu plnění subcíle se

systém vrací nazpět k levému sousedovi před tím již úspěšně splněnému a pokouší se o jeho opětné plnění. Přitom dochází k uvolnění všech proměnných vázaných od předchozí unifikace tohoto subcíle.

Jestliže levý soused neexistuje, je o neúspěchu informován nadřízený cíl, který se pokouší o úspěch u následující klauzule až do okamžiku jeho splnění nebo nesplnění (v případě, že se v databázi nenachází další unifikovatelná klauzule).

Celý tento proces hledání nových alternativ řešení se nazývá **navrácení**. Uveďme si příklad: nechť je k dispozici databáze

otec(petr, pavel)
otec(jan, marie)
otec(jan, ivan)
muž(pavel)
muž(ivan)
žena(marie)
syn(X, Y) :– otec(Y, X), muž(X)

Dotaz nechť zní

?- syn(X, jan)

vyjádřeno slovně „Kdo je synem Jana?“.

Postup systému:

- cíl *syn(X, jan)* se unifikuje s poslední klauzulí databáze, tedy s hlavou pravidla. *X* je nevázaná proměnná na rozdíl od *Y*, která zastupuje objekt *jan* pro všechny její výskyty v dané klauzuli

- splnění hlavy pravidla je podmíněno splněním jeho těla, proto se systém pokouší o důkaz pravdivosti prvého subcíle ve formě *otec(jan, X)*. Při unifikaci uspěje druhá klauzule v pořadí – v tomto případě *otec(jan, marie)*, což tedy vede k instalaci *X* na objekt *marie.* Subcíl tedy uspěje a systém pokračuje plněním dalšího subcíle *muž(marie)*

- vzhledem k tomu, že se v databázi nevyskytuje fakt potvrzující tento subcíl (Marie je žena a ne muž), dochází k procesu navrácení s následujícími důsledky: uvolňuje se proměnná *X* navázaná během předchozí unifikace subcíle *otec(jan, X)* a dochází k novému pokusu opět tento subcíl unifikovat ve zbývající části databáze. Hned následující – v pořadí třetí – klauzule *otec(jan, ivan)* uspěje, čímž dojde k nové instalaci proměnné *X* na objekt *ivan* a následující subcíl *muž(ivan)* se pokouší o své plnění.

- vzhledem k existenci faktu *muž(ivan)* v databázi tento cíl uspěje a je předložen výsledek

X=ivan.

Řízení navrácení

Hlavním nástrojem navrácení je **řez** reprezentovaný v programu symbolem *!*. Řez je v podstatě cílem, který je okamžitě splněn. Pokud však dojde k procesu navrácení, znemožňuje opětovné plnění subcílů nacházejících se nalevo od něj.

Jsou-li tedy k dispozici pravidla

hlava :– a, b, dalsi, c, d
dalsi :– o, p, !, q, r, s
dalsi :– x, y

a dojde ke splnění subcílů *a, b* systém se pokouší plnit pravidlo *dalsi* a uspějí-li také *o, p* pak dojde k okamžitému plnění řezu *!*.

Pokud ovšem následující subcíl *q* neuspěje, řez zabrání znovu plnění všech subcílů nacházejících se nalevo od něj, tedy i samotného cíle *dalsi*. To znamená, že tento cíl neuspívá, o čemž je informován nadřazený cíl *hlava*.

Význam druhé klauzule *dalsi* spočívá v řešení situace, kdy neuspěje již *o* nebo *p* a dojde k procesu navrácení. Tento není ničím ovlivněn, neboť řez se nachází až za neúspěšnými cíli.

Další možnost řídit proces navrácení poskytuje vestavěný predikát *fail*. Jde o cíl vždy nesplněný, který se k vyvolání procesu navrácení nejvíce používá. Uveďme příklad:

syn(X, Y) :– otec(Y, X), muž(X), write(X), fail
syn(_ , _)

Takový výraz povede k umělému vyvolání navrácení a tedy k předložení všech možností odvoditelných v dané databázi. Kdyby tento predikát neměl svoji druhou realizaci v klauzuli *syn(_ , _)* , pak by po prověření a předložení všech možností došlo k jeho neúspěchu. V uvedeném případě však dojde k navrácení a tedy i k plnění druhé klauzule, která vždy uspěje.

Dalším predikátem pro řízení procesu navrácení je predikát *repeat*, který je vždy splněn a je nekonečněkrát opět splnitelný. Jeho realizace je dána klauzulemi

repeat
repeat :– repeat

Posledním predikátem pro řízení procesu navrácení je predikát *true,* který je vždy splněn, ale není opětovně splnitelný.

Negace a disjunkce cílů

Smysl negace v systému PROLOG spočívá v tom, že negace cíle

not(Cíl)

uspěje tehdy, pokud neuspěje vlastní *Cíl* a naopak. Výsledný úspěch negace tedy neznamená pouze fakt, že *Cíl* při svém plnění neuspěl, ale také situaci, že vůbec nebyl

v databázi obsažen (druhý případ by tedy spíše odpovídal výsledku *Nevím* než úspěšné negaci).

Disjunkce cílů je realizována pomocí středníku *;* nebo, což je výhodnější z hlediska čitelnosti, pomocí alternativní formule. To tedy znamená, že predikát *rodič* lze realizovat buďto jednou klauzulí

rodič(X, Y) :– otec(X, Y), matka(X, Y)

nebo dvěma alternativními klauzulemi

rodič(X, Y) :– otec(X, Y)
rodič(X, Y) :– matka(X, Y)

Operátorový zápis struktury

V případě, že struktura je tvořena funktorem a jedním, maximálně dvěma argumenty, je možno použít tzv. **operátorového zápisu** struktury. V něm je funktor deklarován pomocí vestavěného predikátu

op(Precedence, Pozice, Název)

Precedence je celé číslo, které určuje, v jakém pořadí se mají jednotlivé operátory vyhodnocovat. Čím je toto číslo větší, tím je slabší vazba operátoru na operandy. Je-li např. definován operátor + s precedencí 31 a operátor ∗ s precedencí 21 pak výraz

a ∗ b + c ∗ d

bude vyhodnocován jako

(a ∗ b) + (c ∗ d).

Pozice je atom vyjadřující pozici operátoru *f* vzhledem k operandům *(x, y)* a má následující tvary

f x - *prefixový*
x f - *postfixový*
x f x - *infixní, který se neřetězí*
x f y - *infixní, zpraa asociativní*
x f y f z = x f (y f z)
y f x - *infixní, zleva asociativní*
x f y f z = (x f y) f z

Název je atom reprezentující jméno struktury.

Budeme-li definovat predikát

otec(petr, pavel)

pomocí predikátu *op* jako

op(100, xfx, je_otcem)

můžeme používat čitelnějšího zápisu

petr je_otcem pavla.

Modifikace programu

Modifikace programu (databáze) v průběhu jeho vykonávání je jeden ze silných prostředků PROLOGu. K tomuto účelu slouží následující vestavěné predikáty:

assert (klauzule)
asserta (klauzule)
assertz (klauzule)
retract (klauzule)

kde první dva slouží k přidání klauzule na začátek databáze, třetí k přidání klauzule na konec databáze a čtvrtý k odstranění první vyskytující se shodné klauzule z databáze.

Databázi programu z paměti počítače lze načíst dvěma vestavěnými predikáty, a to:

consult (jméno_souboru)
reconsult (jméno_souboru)

kde první přidává načítané klauzule ke stávající databázi programu a druhá nahrazuje predikáty stejného jména nově načítanými. Opačný predikát k oběma předchozím je

save (jméno_souboru)

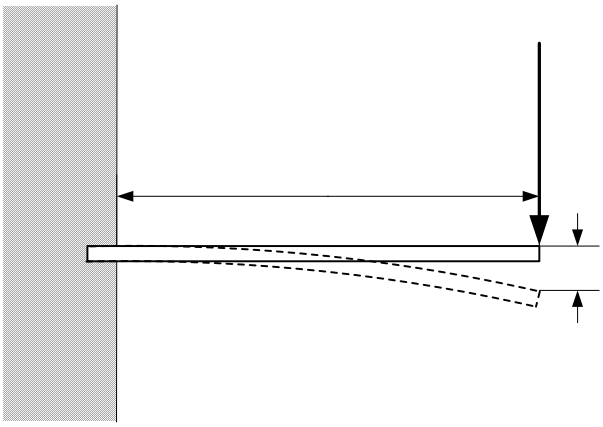
který databázi programu ukládá do paměti.

Systém PROLOG je vybaven řadou dalších vestavěných predikátů (*read, write, ...*) které slouží k různým operacím usnadňujících jeho využívání. V našem případě výše uvedený popis stačí k vytvoření základní představy o výstavbových principech jazyka a pochopení následujících příkladů jeho použití. Úplný seznam predikátů lze získat např. v [3].

3.3.1 Technická úloha v PROLOGu

Uvedeme nyní příklad jednoduché praktické úlohy, který demonstruje, jak systém PROLOG vnáší do aplikací nové postupy, které lze charakterizovat jako inteligentní.[5]

Řešme technickou úlohu výběru profilu vetknutého ocelového nosníku, na konci zatíženého silou F, při zadané délce L a povoleném maximálním průhybu Y_{max} (Obr. 3.1).



Obr. 3.1

Tento cíl budeme formulovat predikátem

výběr_profilu (F, L, Ymax)

kde argumenty budou v dotazu vázány na konkrétní vstupní hodnoty. Budeme předpokládat, že k výběru máme ve skladu k dispozici profily I8, I10 a I12 s odpovídajícími momenty setrvačnosti 77.8, 171.0 a 328.0 cm^4 a profily U14 a U16 s momenty 605.0 a 925.0 cm^4. Tato fakta budeme reprezentovat klauzulemi

profil (i8, 77.8)
profil (i10, 171.0)
profil (i12, 328.0)
profil (u14, 605.0)
profil (u16, 925.0)

Vztah pro výpočet velikosti průhybu Y v [cm] má tvar

$$Y = F * L * L * L / (3 * 21000000 * I)$$

Pokud konstruktér bude tento výpočet provádět ne jednou, ale několikrát, bude mít zřejmě snahu získané výsledky někam uložit a před každým dalším výpočtem bude zjišťovat, zda již pro dané vstupy tento výpočet neprováděl. Systém PROLOG umožňuje imitaci takového (inteligentního) postupu člověka následujícím způsobem.

K archivaci výsledků bude sloužit predikát

výsledek (F, L, Ymax, Profil)

přičemž počet výskytů klauzulí *výsledek* v databázi bude odpovídat počtu různých výpočtů, které byly provedeny.

Přistupme nyní k formulaci programu tak, aby imitoval (inteligentní) činnost konstruktéra:

```
profil ( i8, 77.8 )
profil ( i10, 171.0 )
profil ( i12, 328.0 )
profil ( u14, 605.0 )
profil ( u16, 925.0 )

výběr_profilu ( F, L, Max )  :–
        výsledek (F, L, Max, P ), ! ,
        write („Úloha již byla řešena, použij:“ P )
výběr_profilu ( F, L, Max )  :–
        profil ( P, I ) ,
        Y is F∗L ∗L∗L / (3∗21000000∗I )
        Y <=Max, ! ,
        assert ) výsledek ( F, L, Max, P ))
        write („Nový výpočet, použij:“ P )
výběr_profilu ( _ , _ , _ )  :–
        write („Odpovídající profil není na skladu“ )
```

Výklad a komentář k programu si každý student na základě dosavadního výkladu pokusí formulovat sám. V případě problémů budou dotazy probrány a vysvětleny v rámci konzultací nebo komentář může rychle najít v [8].

3.3.2 Expertní systém v PROLOGu

Jak již bylo řečeno v předcházejícím výkladu, hlavními součástmi expertního systému jsou báze znalostí a inferenční mechanizmus. Uveďme předně způsob formulace znalostí a báze dat.

Efektivní formou reprezentace expertních znalostí jsou podmíněná pravidla typu

jestliže *podmínka* pak *důsledek*

Podmínka může být tvořena logickou strukturou (konjunkce, disjunkce, negace) a závorkami, vyjadřujícími prioritu vyhodnocování.

Důsledek je výrok, jehož hodnota vyplývá z pravdivosti tvrzení předpokladu – podmínky.

Báze dat je pak budována na základě odpovědí uživatele na dotazy systému. Na základě odpovědí jsou deklarována fakta, začleňovaná do databáze ve formě predikátů *poz_fakt* v případě, že je platný nebo *neg_fakt* v případě opačném ve formě:

poz_fakt (položka_báze_dat)
neg_fakt (položka_báze_dat)

Abychom přiblížili formulaci znalostí přirozenému jazyku, jsou vytvořeny nové operátory pomocí vestavěných predikátů *op* takto:

op (200 , f x , cil)
op (200 , f x , dotaz)
op (200 , f x , jestlize)
op (200 , f x , pak)
op (200 , f x , a)
op (200 , f x , nebo)

přičemž

cil vyjadřuje co má být inferenčním mechanizmem odvozeno

dotaz slouží inferenčnímu mechanizmu pro identifikaci dotazů ES na uživatele

jestliže a *pak* tvoří vlastní tvar pravidla báze znalostí

a a *nebo* slouží k vytváření logických výrazů.

Vytvořme nyní demonstrační báze znalostí, určené k rozpoznávání druhů zvířat takto [8]:

cíl savec
cíl ptak
cíl bylozravec
cíl selma
cíl dravec

Predikát *Cíl* vyjadřuje jednotlivá tvrzení o druzích zvířat, které mají být inferenčním mechanizmem odvozeny na základě těchto expertem definovaných pravidel:

jestliže pije_mleko pak savec
jestliže neg(savec) a (ma_peri nebo leta) pak ptak
jestliže savec a (ma_drapy nebo zere maso) pak selma
jestliže ptak a (ma_drapy nebo zere maso) pak dravec
jestliže savec a neg(selma) pak bylozravec

a dále na základě plnění následujících dotazů (báze dat uživatele):

dotaz pije_mleko
dotaz ma_peri
dotaz leta
dotaz ma_drapy
dotaz zere_maso

Odpovědi uživatele na tyto dotazy pak vedou k postupnému začleňování predikátu *poz_fakt* nebo *neg_fakt* do databáze programu.

Vlastní postup odvozování (inferenční mechanizmus) používá principu zpětného řetězení, je realizován predikátem *hledej* a je složen z následujících klauzulí:

```
hledej ( Cil ) :–
        poz_fakt ( Cil ) , ! .
hledej ( Cil ) :–
        neg_fakt ( Cil ) , !
        fail .
hledej ( neg ( Cil ))  :–
        hledej ( Cil ) , ! , fail ; ! .
hledej ( Cil ) :–
         dotaz ( Cil ) , ! ,
        dotazuj (Cil ).
hledej ( Cil ) :–
        jestliže Podmínka pak Cil ,
        hledej ( Podmínka ).
hledej ( Cil1 a Cil2 ) :–
        hledej ( Cil1 ) ,
        hledej ( Cil2 ) .
hledej ( Cil1 nebo Cil2 ) :–
        hledej ( Cil1 ) , ! ;
        hledej ( Cil2 ) .
```

První a druhá klauzule

řeší situaci, kdy ověřovaný cíl se nachází v databázi ve formě pozitivního faktu, nebo naopak ve formě faktu negativního.

Třetí klauzule

ověřuje platnost negace cíle. Pokud platí vlastní cíl, nemůže platit jeho negace a odvozování končí neúspěchem, jinak dochází k plnění cíle.

Čtvrtá klauzule

ověřuje - dříve než se predikát *hledej* pokusí o řetězení – zda cíl není listovou hypotézou a není tudíž nutno položit dotaz (pomocí predikátu *dotazuj*) na jeho platnost uživateli.

Pátá klauzule

realizuje vlastní proces zpětného řetězení, kdy se ztotožňuje ověřovaný cíl s důsledkem pravidla a inferenční mechanizmus postupuje dále v ověřování, tentokrát pro cíl reprezentovaný podmínkou pravidla.

Šestá a sedmá klauzule

řeší situaci, kdy cíl je tvořen konjunkcí nebo disjunkcí dvou subcílů.

Řízení dotazování uživatele provádí predikát *dotazuj*, jehož hlavním úkolem je položení dotazu a začlenění splněného (nesplněného) faktu do databáze programu.

```
dotazuj ( Cil ) :–
        write ( " Je splneno tvrzeni " ) [ a / n] " , Cil , " " ) ,
        read1n ( Odp ) ,
        Odp = " a " , ! ,
        assert ( poz_fakt ( Cil )) .
dotazuj ( Cil ) :–
        assert ( neg_fakt ( Cil )) ,
        fail .
```

Celý program expertního systému má následující podobu:

```
/* Spuštěni IM */
        spuštěni :–
                retractall ( poz_fakt ( _ )) ,
                retractall ( neg_fakt ( _ )) ,
                retractall ( poz_fakt ( _ )) ,
                reconsult ( v_es_ dat ) ,
                cil Vysledek ,
                hledej_reseni ( Vysledek ) ,
                write ( " Dalsi cil ? [ a / n]: " ) ,
                read1n ( Odp ) ,
                Odp = "n" ,
                write ( " To jest vse ! " ) .
        hledej_reseni ( Cil ) :–
                hledej ( Cil ) , n1 ,
                write ( " Cil " , Cil , " je splnen ! " ) ,
                n1 , n1 , ! .


/* Realizace inferencniho mechanizmu */

        hledej ( Cil ) :–
                poz_fakt ( Cil ) , ! .
        hledej ( Cil ) :–
                neg_fakt ( Cil ) , !
                fail .
        hledej ( neg ( Cil ))  :–
                hledej ( Cil ) , ! , fail ; ! .
```

```
hledej ( Cil ) :–
        dotaz ( Cil ) , ! ,
        dotazuj (Cil ).
hledej ( Cil ) :–
        jestliže Podmínka pak Cil ,
        hledej ( Podmínka ).
hledej ( Cil1 a Cil2 ) :–
        hledej ( Cil1 ) ,
        hledej ( Cil2 ) .
hledej ( Cil1 nebo Cil2 ) :–
        hledej ( Cil1 ) , ! ;
        hledej ( Cil2 ) .

/* Dotaz na platnost dotazu */

dotazuj ( Cil ) :–
        write ( " Je splneno tvrzeni " ) [ a / n] " , Cil , " " ) ,
        read1n ( Odp ) ,
        Odp = " a " , ! ,
        assert ( poz_fakt ( Cil )) .
dotazuj ( Cil ) :–
        assert ( neg_fakt ( Cil )) ,
        fail .

retractall ( X ) :–
        retract ( X ) , fail .
retractall ( _ ) .
```

Po splnění klauzulí procesu spuštění je zahájena konzultace s uživatelem, kterému je předložen první dotaz

Je splneno tvrzeni pije_mleko [a/n]

Odpovíme-li *a* – ano inferenční mechanizmus postupuje v odvozování a následuje předložení jeho prvního výsledku

Cíl savec je splněn !

Vynucení dalšího hledání řešení, která splňují fakta, vede k dalšímu dotazu

Je splneno tvrzeni ma_drapy [a/n]

Na tento odpovíme *n* – ne, stejně jako na dotaz následující

Je splneno tvrzeni zere_maso [a/n]

Což vede k předložení cíle

Cíl bylozravec je splněn !

Další vyhledávání cílů pak končí hlášením

To jest vše !

Základy programování v PROLOGu - shrnutí

Řešení problémů nástroji umělé inteligence používajícího jazyka PROLOG je uvedeno detailnějším popisem postupu programování, který uvádí způsob sestavení databáze programu vycházející z typické formy seznamu, uvádí základní principy programování jako jsou unifikace a vázání proměnných, princip navrácení a způsob jeho řízení, negaci a disjunkci cílů, operátorový zápis struktury a pojem modifikace programu. Uvádí příslušné predikáty jazyka a ilustruje výklad praktickým příkladem programu pro řešení vědecko-technického výpočtu a jednoduchého expertního systému.

Kontrolní otázky:

1. Jaký je význam SEZNAMU jako formy programu PROLOGu a jak může být SEZNAM realizován?
2. Dovedete vysvětlit princip procedur *Rekurze*, *Unifikace a vázání proměnných*, *Navrácení* a *Modifikace programu* v jazyku PROLOG?

Otázka k zamyšlení:

1. Jak je možno v jazyku PROLOG postupovat při řešení problému v kap.3.3 aniž bychom využili princip navrácení?

Úkoly k zamyšlení:

Promyslete konstrukci programu vlastní jednoduché úlohy napsané v jazyku PROLOG.

3.4 Programovací jazyk LISP

Jak již bylo řečeno, jazyk LISP [9] je prostředek pro vytváření programů pro zpracování nenumerických objektů organizovaných do seznamové struktury (LISP – LIst PRocessor). Výpočtový proces je v jazyku LISP zapsán pomocí **kompozice funkcí**. Svoji syntaxi i sémantiku odvozuje od matematické teorie rekurzivních funkcí. Programování pomocí kompozice funkcí se nazývá funkcionální programování a jazyk LISP je proto **funkcionální jazyk**.

Program i všechny podprogramy v LISPu jsou funkcemi, tj. mají několik argumentů a vždy vracejí určitou hodnotu jako výsledek. Výsledek vyhodnocení nějaké funkce bývá argumentem další funkce atd. až po hlavní program, tj. po funkci na nejvyšší (nulté) úrovni, jejíž výsledek je výsledkem celého výpočtu.

Pojem funkce je centrálním pojmem LISPu. Objekty se kterými funkce pracuje (vstupní údaje) zadáváme prostřednictvím parametrů a nazýváme je **argumenty.** Protože zápisy funkcí vystupují jako argumenty, vyhodnocení funkce začíná vyhodnocováním argumentů. To je základní princip, jímž se řídí provádění programu v LISPu.

Předností jazyka LISP je jeho jednoduchost a logická jasnost. Je vhodný především pro programování široké škály nenumerických úloh. Zájemci o hlubší informace o vlastnostech a použití jazyka LISP naleznou potřebné informace např. v [9] nebo [10].

Programovací jazyk LISP - shrnutí

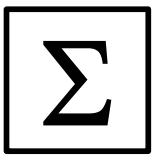
Programovací jazyk umělé inteligence LISP je prostředek pro zpracování údajů organizovaných do seznamové struktury. Využívá zápisu ve formě kompozice funkcí a je typickým jazykem funkcionálním. Jeho předností je jednoduchost a jasnost.

Kontrolní otázky:

1. Které jsou základní společné rysy jazyka LISP a jazyka PROLOG?

Korespondenční úkoly:

Napište jádro programu řešení tecnické úlohy v jazyce PROLOG
Navrhněte expertní systém pro řešení jednoduchého problému a rozepište postup jeho programového zpracování v jazyku PROLOG



Shrnutí obsahu kapitoly

Umělá inteligence přinesla zásadní změnu v nárocích na programování, která spočívá v nutnosti přechodu od zpracování dat ke zpracování informací (znalostí). Tento požadavek se odrazil v odklonu od klasického programování procedurálního a přechodu na programování deklarativní. Jeho základem je oddělení zákonitostí řešení problému od vlastních řešících procesů. Charakter programů umělé inteligence vyžaduje efektivní práci s daty v symbolickém tvaru. Proto vznikly speciální programovací jazyky, např. jazyk LISP nebo PROLOG. Kapitola uvádí principy jazyka PROLOG jako reprezentanta jazyků logického programování. Řešení problémů nástroji umělé inteligence používajícího jazyka PROLOG je uvedeno detailnějším popisem postupu programování, uvádí příslušné predikáty jazyka a ilustruje výklad praktickým příkladem programu pro řešení vědecko-technického výpočtu a jednoduchého expertního systému. Programovací jazyk umělé inteligence LISP je prostředek pro zpracování údajů organizovaných do seznamové struktury, využívá zápisu ve formě kompozice funkcí a je typickým jazykem funkcionálním.

Doporučená literatura

[1] Popper,M., Kelemen,J.: Expertné systémy, ALFA Bratislava, 1988
[2] Minsky,M.: Why people think computers can´t, MIT Technology Review 3, 86, 1983
[3] Polák,J.: PROLOG, Grada Praha, 1992
[4] Kowalski,R.: Logic for Problem Solving, North-Holland, Amsterdam, 1979
[5] Clocksin,W.F., Mellish,C.S.: Programming in PROLOG, Springer-Verlag, Berlin, 1981
[6] Bratko,L.: Prolog Programming for Artificial Intelligence, Addison-Wesley, 1986
[7] Havel,I.M.: Robotika, SNTL Praha, 1983
[8] Vondrák,I.: Umělá inteligence, UP Olomouc, 1990
[9] Molnár,L., Návrat,P.: Programovanie v jazyku LISP, ALFA Bratislava, 1988
[10] Siklóssy, L.: Let´s Talk LISP, Englewood Cliffs, Prentice-Hall, 1976

4 PRAVDĚPODOBNOSTNÍ EXPERTNÍ SYSTÉMY

V této kapitole se dozvíte:

- Jak je pojato řešení problémů s využitím modifikovaných pravděpodobností platnosti produkčních pravidel
- Způsob řešení pravděpodobnostního přístupu v diagnostickém modelu typu MYCIN a EMYCIN
- Způsob řešení pravděpodobnostního přístupu v diagnostickém modelu typu PROCPECTOR

Po jejím prostudování byste měli být schopni:

- Vysvětlit princip pravděpodobnostních metod řešení expertních systémů
- Popsat pravděpodobnostní modely typu MACIN, EMYCIN a PROSPECTOR
- Vytvořit jednoduchý pravděpodobnostní diagnostický expertní systém

Klíčová slova této kapitoly:
Apriorní pravděpodobnost, aposteriorní pravděpodobnost, míry subjektivní pravděpodobnosti, pravděpodobnostní model MYCIN, pravděpodobnostní model EMYCIN, pravděpodobnostní model PROSPECTOR

Doba potřebná ke studiu: 3 hodiny

Průvodce studiem

Prvními celosvětově úspěšnými diagnostickými expertními systém, jejichž principy jsou používány dodnes, byly systémy založené na modifikovaných přístupech využití matematické pravděpodobnosti. Jsou to ad-hoc techniky, používající subjektivní apriorní pravděpodobnosti platnosti produkčních pravidel jako hypotéz. Seznámíme se s přístupy takových dnes již prototypových pravděpodobnostních modelů MYCIN a PROSPECTOR. Na příkladech řešení jednoduchých diagnostických expertních systémů si ukážeme jejich praktické použití

4.1 Model pro práci s neurčitostí typu MYCIN

Expertní systém MYCIN [1] je jedním z prvních celosvětově úspěšných expertních systémů, jehož řešení se stalo (spolu v následující podkapitole uvedeným expertním systémem PROSPECTOR) prototypem pro řešení celé řady dalších úspěšných systémů.

V systému MYCIN jsou znalosti vyjádřeny pomocí produkčních pravidel typu

$$\langle \text{antecedent} \rangle \Rightarrow \langle \text{konsekvent} \rangle$$

které jsou vyjádřeny symboly podmíněné platnosti hypotézy (důsledku) H za předpokladu splnění evidence (předpokladu) E

$E \Rightarrow H$

Neurčitost pravidla je v systému MYCIN vyjádřena pomocí **míry důvěry** v platnost hypotézy *H* označené *MB(H,E)* a současně **mírou nedůvěry** označené *MD(H,E)*. Tyto velmi důležité míry jsou definovány vztahy

$$MB(H,E) = \frac{P(H \mid E) - P(H)}{1 - P(H)} \tag{4.1}$$

$$MD(H,E) = \frac{P(H) - P(H \mid E)}{P(H)} \tag{4.2}$$

kde *P(H)* je **apriorní pravděpodobnost**, kterou zadává expert a $P(H \mid E)$ je **podmíněná pravděpodobnost** platnosti hypotézy *H* za předpokladu splnění evidence *E*.

Při návrhu takového systému je podstatné, jakým způsobem je řešen problém vzrůstu důvěry (resp. nedůvěry) v platnost hypotézy *H* na základě potvrzení evidence *E*.

Jestliže splnění *E* vede ke vzrůstu důvěry v *H*, platí

$P(H \mid E) > P(H)$,

velikost míry důvěry je vypočítávána podle vztahu (4.1) a míra nedůvěry $MD(H,E) = 0$. Míra důvěry *MB(H,E)* tak vyjadřuje přírůstek pravděpodobnosti *H*, získaný pomocí evidence *E,* vztažený k počáteční (apriorní) nedůvěře v *H,* tj. $1 - P(H)$.

Jestliže ale splnění evidence *E* vede k poklesu důvěry v hypotézu *H*, platí

$P(H) > P(H \mid E)$,

míra nedůvěry *MD(H,E)* je dána vztahem (4.2) a $MB(H,E) = 0$. Míra *MD(H,E)* tak vyjadřuje pokles pravděpodobnosti hypotézy *H*, získaný pomocí evidence *E*, vztažený k počáteční (apriorní) důvěře v *H*, tj. *P(H)*.

Míry *MB* a *MD* nabývají hodnot z intervalu <0, 1>. Ze skutečnosti, že určité jediné pravidlo nemůže současně hypotézu podporovat a současně ji vyvracet, platí

a) když $MB > 0$, tak $MD = 0$
b) když $MD > 0$, tak $MB = 0$.

V případě, že pro určité pravidlo platí

$MB = MD = 0$,

potom pravidlo hypotézu ani nepotvrzuje, ani nevyvrací.

Velmi důležitým a typickým kritériem systému MYCIN je **činitel jistoty** *CF* definovaný vztahem

$$CF(H,E) = MB(H,E) - MD(H,E) \tag{4.3}$$

který spojuje míru důvěry a míru nedůvěry do jednoho čísla. Činitel jistoty *CF* nabývá hodnot z intervalu <-1, +1>. Platí tato pravidla:

a) když evidence *E* zvyšuje důvěru v *H*, tak $CF > 0$
b) když evidence *E* snižuje důvěru v *H*, tak $CF < 0$.

Ke každému pravidlu $E \Rightarrow H$ je tak v systému MYCIN přiřazen činitel jistoty *CF(H,E)*. Pokud je kladný, je roven *MB(H,E)*. Pokud je záporný, je v absolutní hodnotě roven *MD(H,E)*.

Pro úplnost uveďme, že varianta systému MYCIN nazvaná EMYCIN [2] používá modifikovaného vztahu

$$CF = \frac{MB - MD}{1 - \min(MB, MD)} \tag{4.4}$$

Velmi důležitá je skutečnost, že v praxi nemusí být evidence $E_1 \Rightarrow H$ splněna s jistotou. Na základě konkrétního případu (pozorování) *E´* můžeme odhadnou (popsat) naším **odhadem** míry jistoty $CF(E_1,E')$.

Označíme-li $MB(H,E_1)$, resp. $MD(H,E_1)$ míru důvěry resp. míru nedůvěry v *H* je-l splněna evidence E_1, můžeme výslednou míru důvěry (nedůvěry) vypočítat ze vztahu

$$MB(H, E') = MB(H, E_1).\max\{0, CF(E_1, E')\} \tag{4.5}$$

$$MD(H, E') = MD(H, E_1).\max\{0, CF(E_1, E')\} \tag{4.6}$$

Dalším důležitým problémem je řešení situace, kdy více pravidel, např.

$E_1 \Rightarrow H$
$E_2 \Rightarrow H$,

které mají stejnou hypotézu *H*, působí současně a je třeba stanovit výslednou míru jistoty v platnost této hypotézy. Předpokládáme-li, že ani v jednom z takových pravidel není hypotéza *H* zcela potvrzena nebo zcela vyvrácena, vypočítáme výsledné míry podle vztahů

$$MB(H, E_1 \,\&\, E_2) = MB(H, E_1) + MB(H, E_2) - MB(H, E_1).MB(H, E_2) \tag{47}$$

$$MD(H, E_1 \,\&\, E_2) = MD(H, E_1) + MD(H, E_2) - MD(H, E_1).MD(H, E_2) \tag{4.8}$$

Výsledný činitel jistoty pak vypočítáme podle vztahu (4.3). Vzhledem k symetrii vztahů platí

$CF(H,E_1 \,\&E_2) = CF(H,E_2 \,\&E_1)$.

Poznamenejme, že v systému EMYCIN platí alternativní vztah

$$CF(H, E_1 \,\&\, E_2) = f(CF(H, E_1), CF(H, E_2)) \tag{9}$$

kde f je definováno jako

$$f(x,y) = \begin{cases} x + y - xy & \text{pro } xy \geq 0 \\ \dfrac{x+y}{1 - \min\{\text{abs}(x), \text{abs}(y)\}} & \text{v ostatních případech} \end{cases}$$

Posledním problémem, který je z praktického hlediska použití systému MYCIN třeba řešit, je případ konjunkce či disjunkce hypotéz H_1 a H_2. Systém MYCIN zde užívá vzathů

$$MB(H_1 \,\&\, H_2, E) = \min\{MB(H_1, E), MB(H_2, E)\} \tag{4.10}$$

$$MD(H_1 \,\&\, H_2, E) = \max\{MD(H_1, E), MD(H_2, E)\} \tag{4.11}$$

$$MB(H_1 \vee H_2, E) = \max\{MB(H_1, E), MB(H_2, E)\} \tag{4.12}$$

$$MD(H_1 \vee H_2, E) = \min\{MD(H_1, E), MD(H_2, E)\} \tag{4.13}$$

Uveďme nyní jednoduchý příklad. Nechť je např. dáno pravidlo tvaru

$$(E_1 \,\&\, E_2) \vee E_3 \Rightarrow H \tag{4.14}$$

s mírou jistoty $CF = c$. Neurčitost předpokladů určí uživatel na základě pozorování konkrétní situace pomocí jím stanovených hodnot $MB(E_1,E')$, $MB(E_2,E')$, $MB(E_3,E')$, $MD(E_1,E')$, $MD(E_2,E')$ a $MD(E_3,E')$, kde E' je jeho pozorování. Míru jistoty v platnost hypotézy H která má tvar (4.14) vypočítáme s použitím vztahů (4.3), (4.5) a (4.6) jako

$$\begin{aligned} CF(H,E') &= MB(H,E') - MD(H,E') = \\ &= MB(H,(E_1 \,\&\, E_2) \vee E_3).\max\{0, CF((E_1 \,\&\, E_2) \vee E_3, E')\} - \\ &\quad - MD(H,(E_1 \,\&\, E_2) \vee E_3).\max\{0, CF((E_1 \,\&\, E_2) \vee E_3, E')\} = \\ &= c.\max\{0, CF((E_1 \,\&\, E_2) \vee E_3, E')\} \end{aligned}$$

Výraz $CF(E_1 \,\&\, E_2) \vee E_3, E'$ rozložíme podle (4.3) a postupnou aplikací vztahů (4.10) – (4.13) dostaneme závislost na uvedených mírách $MB(E_1,E')$, $MB(E_2,E')$, $MB(E_3,E')$, $MD(E_1,E')$, $MD(E_2,E')$ a $MD(E_3,E')$. Tyto hodnoty spolu s hodnotou c známe a proto můžeme vypočítat celkovou $CF(H,E')$.

4.1.1 Expertní systém typu MYCIN

Pro studenty-motoristy uvedeme příklad skutečné diagnostiky spalovacího čtyřtaktního motoru. Uvažujme (fragment) báze znalostí ve tvaru čtyř pravidel:

$\text{IF}\langle E_1 : \text{Klepou ventily po } p\text{řidání plynu}\rangle$
$\text{THEN}\langle H : \text{Špatně se}\check{r}\text{ízený } p\text{ředstih}\rangle$
$\text{WITH}\langle MB(H, E_1) = 0{,}8; MD(H, E_1) = 0\rangle$

$\text{IF}\langle E_2 : \text{Motor střílí do výfuku}\rangle$
$\text{THEN}\langle H : \text{Špatně seřízený předstih}\rangle$
$\text{WITH}\langle MB(H, E_2) = 0{,}92; MD(H, E_2) = 0\rangle$

$\text{IF}\langle E_3 : \text{Malá spotřeba}\rangle$
$\text{THEN}\langle H : \text{Špatně seřízený předstih}\rangle$
$\text{WITH}\langle MB(H, E_3) = 0; MD(H, E_3) = 0{,}9\rangle$

$\text{IF}\langle E_4 : \text{Malá akcelerace}\rangle$
$\text{THEN}\langle H : \text{Špatně seřízený předstih}\rangle$
$\text{WITH}\langle MB(H, E_4) = 0{,}86; MD(H, E_4) = 0\rangle$

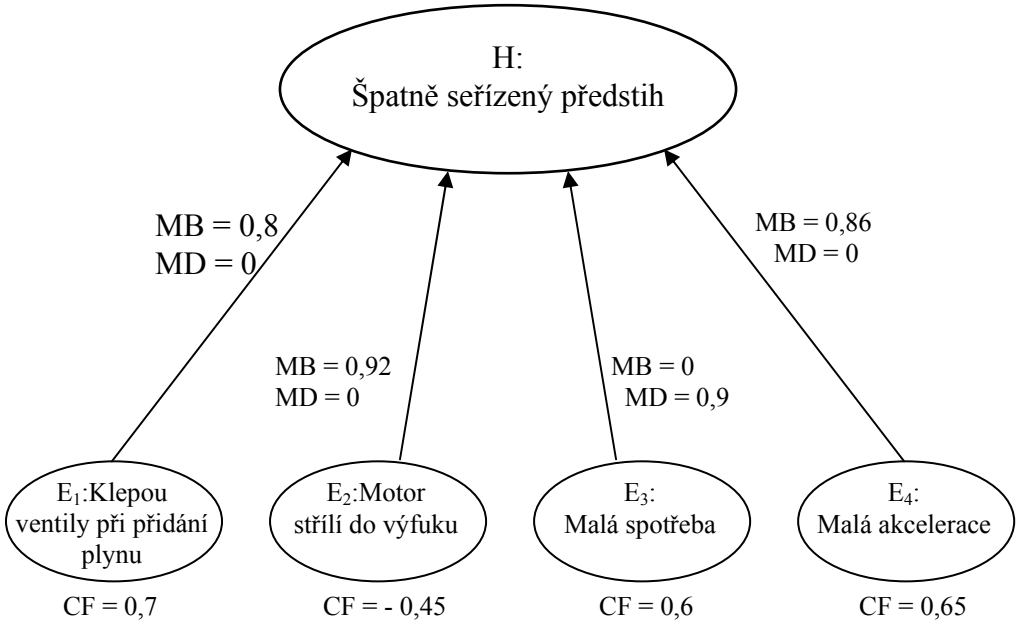
Tato pravidla můžeme překreslit do tvaru inferenční sítě na Obr.4.1.

Předpokládejme nyní, že uživatel posoudí projevy diagnostikovaného motoru a ohodnotí platnost evidencí E_1 až E_4 postupně takto:

$CF(E_1, E') = 0{,}7$,
$CF(E_2, E') = -0{,}45$,
$CF(E_3, E') = 0{,}6$,
$CF(E_4, E') = 0{,}65$,

Pro ohodnocení první evidence platí

$MB(H, E_1') = MB(H, E_1).\max\{0, CF(E_1, E')\} = 0{,}8 \,.\, 0{,}7 = 0{,}56$
$MD(H, E_1') = 0$
$CF(H, E_1') = 0{,}56$



Obr. 4.1

Po určení $CF(E_2,E')$ = -0,45 lze ihned podle vztahů (4.5) a (4.6) vypočítat

$$MB(H,E_2') = MD(H,E_2') = CF(H,E_2') = 0$$

protože pravidlo nepřispívá k ohodnocení H a proto

$$CF(H,E_1' \,\&\, E_2') = CF(H,E_1') = 0{,}56$$

Dále uživatel odhadl $CF(E_3,E')$ = 0,6 , potom

$$MB(H,E_3') = 0$$
$$MD(H,E_3') = MD(H,E_3)\,.\,\max\{0,CF(E_3,E')\} = 0{,}9\,.\,0{,}6 = 0{,}54$$
$$CF(H,E_3') = -0{,}54$$

$$CF(H,E_1' \,\&\, E_2' \,\&\, E_3') = \frac{CF(H,E_1' \,\&\, E_2') + CF(H,E_3')}{1 - \min\{\mathrm{abs}(CF(H,E_1' \,\&\, E_2')), \mathrm{abs}(CF(H,E_3'))\}} =$$
$$= \frac{0{,}56 - 0{,}54}{1 - 0{,}54} = 0{,}0434$$

V posledním kroku pak vypočteme

$$MB(H,E_4') = MB(H,E_4)\,.\,\max\{0,CF(E_4,E')\} = 0{,}86\,.\,0{,}65 = 0{,}559$$
$$MD(H,E_4') = 0$$
$$CF(H,E_4') = 0{,}559$$

$$CF(H, E_1' \,\&\, E_2' \,\&\, E_3' \,\&\, E_4') = CF(H, E_1' \,\&\, E_2' \,\&\, E_3') + CF(H, E_4') - \\ - CF(H, E_1' \,\&\, E_2' \,\&\, E_3') \,.\, CF(H, E_4') = \\ = 0{,}578$$

Povšimneme si, že pozitivní odpovědi uživatele na dotazy

„Klepou ventily při přidání plynu ?“ a *„Má motor malou akceleraci ?“*

podporovaly podezření, že je

„Špatně seřízený předstih“.

Naopak pozitivní odpověď na otázku

„Má motor malou spotřebu ?“

pak podezření na tuto diagnózu zeslabuje (vzhledem k nenulové *MD* ve třetím pravidle) a negativní odpověď na dotaz

„Střílí motor do výfuku ?“

v našem modelu vůbec ohodnocení hypotézy *H* neovlivní.

Model pro práci s neurčitostí typu MYCIN - shrnutí

Expertní systém MYCIN je jedním z prvních úspěšných expertních systémů. Využívá reprezentace znalostí formou produkčních pravidel. Neurčitost předpokladů je vyjádřena pomocí míry důvěry a míry nedůvěry v platnost pravidla. Tyto míry jsou spojeny do činitele jistoty. Neurčitost dat je stanovena na základě pozorování odhadem míry jistoty v platnost splnění předpokladů. Jsou uvedeny principy variantního modelu typu EMYCIN. Řídicí ad-hoc mechanizmus provede ohodnocení pravidel a stanoví nejpravděpodobnější hypotézu jako závěr. Systém MYCIN je ilustrován praktickým příkladem jednoduchého diagnostického expertního systému.

Kontrolní otázky:

1. V čem spočívá idea MYCINu jako expertního systému s ad-hoc řídicím mechanizmem?
2. Jaký je rozdíl mezi apriorní a aposteriorní pravděpodobností výskytu jevu?
3. Jakými parametry je vyjádřena neurčitost pravidla v systému MYCIN?

Otázka k zamyšlení:

1. Jaký je rozdíl mezi systémy MYCIN a EMYCIN?

Úkoly k zamyšlení:

Pokuste se formulovat jiná možná intuitivní kritéria neurčitosti pravidla než ta, která používá systém MYCIN.

4.2 Model pro práci s neurčitostí typu PROSPECTOR

Druhým světově proslulým a dodnes uplatňovaným přístupem k ad-hoc řešení problému ohodnocování hypotéz pomocí pravidel s neurčitostí podává systém PROSPECTOR [3].

Model s jistými pravidly (bez neurčitosti)

Ukážeme si nejprve, jak systém zpracovává případ, kdy ohodnocení evidence *E* v pravidlech typu

$E \Rightarrow H$

nabývají pouze logických hodnot 0 a 1. Jde o situaci, kdy pracujeme s **modelem bez neurčitosti.**

Předpokládejme tedy, že bylo zjištěno, že evidence *E* je pravdivá. Podle již zmíněného klasického Bayesova vztahu pro dokazování platnosti hypotéz (2.1) [4] platí

$$P(H \mid E) = \frac{P(E \mid H) \,.\, P(H)}{P(E)}$$

kde P(*E*) a P(*H*) jsou apriorní pravděpodobnosti předpokladu *E* resp. hypotézy *H*. Podobně pro negaci *H* platí

$$P(\text{not } H \mid E) = \frac{P(E \mid \text{not } H) \,.\, P(\text{not } H)}{P(E)} \tag{4.15}$$

Dělením rovnic (4.14) a (4.15) dostaneme výraz

$$\frac{P(H \mid E)}{P(\text{not } H \mid E)} = \frac{P(E \mid H)}{P(E \mid \text{not } H)} \cdot \frac{P(H)}{P(\text{not } H)} \tag{4.16}$$

Systém PROSPECTOR používá kritéria **naděje** nebo **pravděpodobnostní šance** *O(H)* a $O(H \mid E)$, kteréžto dostaneme označením pravděpodobnostních poměrů z výrazu (4.16) takto:

$$O(H) = \frac{P(H)}{P(\text{not } H)} = \frac{P(H)}{1 - P(H)} \tag{4.17}$$

$$O(H \mid E) = \frac{P(H \mid E)}{P(\text{not } H \mid E)} = \frac{P(H \mid E)}{1 - P(H \mid E)} \tag{4.18}$$

Dále zavedeme tzv. **míru postačitelnosti** L ve tvaru

$$L = \frac{P(E \mid H)}{P(E \mid \text{not } H)} \tag{4.19}$$

Z uvedených vztahů plyne

$$O(H \mid E) = L \,.\, O(H) \tag{4.20}$$

Je-li veličina *O(H)* apriorní pravděpodobnostní šance, pak $O(H \mid E)$ je šance aposteriorní.

Rovnice (4.20) (která je vlastně poměrným tvarem Bayesova vztahu) říká, že bylo-li zjištěno, že předpoklad E je pravdivý ($P(E) = 1$), potom vypočítáme aposteriorní šanci $O(H \mid E)$ vynásobením apriorní šance mírou postačitelnosti L. Míra postačitelnosti L je přitom kvantitativní ocenění platnosti pravidla a zadává ji expert.

Předpokládejme nyní, že chceme vypočítat hodnotu $O(H \mid \text{not}E)$. K tomu lze odvodit vztah

$$O(H \mid \text{not } E) = \hat{L} \,.\, O(H) \tag{4.21}$$

kde $\hat{L}$ je tzv. **míra nezbytnosti**

$$\hat{L} = \frac{P(\text{not } E \mid H)}{P(\text{not } E \mid \text{not } H)} \tag{4.22}$$

Míra nezbytnosti $\hat{L}$ musí být také zadána expertem, toto číslo nelze vypočítat z L. Rovnice (4.21) přitom říká, jak aktualizovat šanci *O(H)*, jestliže bylo shledáno, že předpoklad E není pravdivý.

$$L = \frac{P(H \mid E)}{1 - P(H \mid E)} \cdot \frac{1 - P(H)}{P(H)}$$

Pravidlo $E \Rightarrow H$ v popisovaném modelu lze chápat jako dvojpravidlo

$$\text{IF } \langle \text{předpoklad } E \rangle \text{ THEN } \langle \text{závěr } H \rangle \text{ WITH } \langle \text{váha } L \rangle$$
$$\text{ELSE } \langle \text{závěr } H \rangle \text{ WITH } \langle \text{váha } \hat{L} \rangle,$$

a lze je tedy zapsat ve formě dvou samostatných pravidel

$$\text{IF } \langle \text{předpoklad } E \rangle \text{ THEN } \langle \text{závěr } H \rangle \text{ WITH } \langle \text{váha } L \rangle$$
$$\text{IF } \langle \text{předpoklad not } E \rangle \text{ THEN } \langle \text{závěr } H \rangle \text{ WITH } \langle \text{váha } \hat{L} \rangle$$

Model s nejistými pravidly (s neurčitostí)

Nyní budeme předpokládat, evidence E v pravidlech $E \Rightarrow H$ nebudou nabývat pouze logických hodnot „pravda“ (tj. pravdivostní hodnoty 1) nebo „nepravda“ (pravdivostní hodnoty 0), nýbrž že jejich platnost bude možno ohodnotit pravdivostními hodnotami z intervalu $\langle 0, 1 \rangle$. Budeme nyní pracovat s **modelem s neurčitostí.**

Předpokládejme tedy, že uživatel např. může pouze např. říci: :

„Jsem si na 70% jist, že E je pravda“.

Tuto jeho výpověď budeme interpretovat jako

$P(E \mid E') = 0{,}7$

kde E' představuje relevantní pozorování. Nyní potřebujeme nalézt vztah pro určení $P(H \mid E')$. Teoreticky platí:

$$\begin{aligned} P(H \mid E') &= P(H, E \mid E') + P(H, \text{not } E \mid E') = \\ &= P(H \mid E, E') \,.\, P(E \mid E') + P(H \mid \text{not } E, E') \,.\, P(\text{not } E \mid E') \end{aligned} \tag{4.23}$$

Vyjděme z tohoto předpokladu: víme-li, že E je pravda nebo nepravda, pozorování E' nepřináší o H žádnou novou informaci. Proto můžeme psát (4.23) ve tvaru

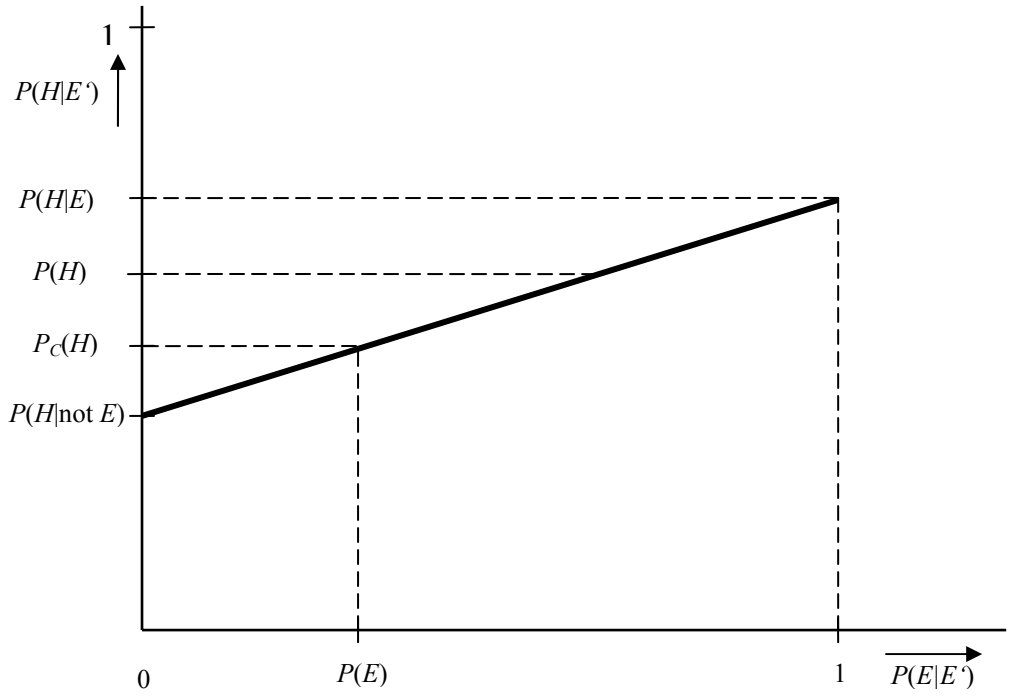
$$\begin{aligned} P(H \mid E') &= P(H, E) \,.\, P(E \mid E') + P(H \mid \text{not } E) \,.\, P(\text{not } E \mid E') = \\ &= P(H \mid \text{not } E) + \left[P(H \mid E) - P(H \mid \text{not } E)\right] . P(E \mid E') \end{aligned} \tag{4.24}$$

Hodnoty $P(H \mid E)$ a $P(H \mid \text{not}E)$ zadává expert přímo, nebo je lze vypočítat z L a $\hat{L}$. Rovnice (4.24) představuje lineární závislost mezi $P(H \mid E')$ a $P(E \mid E')$ a lze ji znázornit přímkou podle Obr. 4.2.

Povšimněme si ale jedné důležité věci: protože expert zadává nezávisle $P(H \mid E)$ a $P(H \mid \text{not}E)$, $P(E)$ a $P(H)$, je přímka podle rovnice (4.24) z matematického hlediska přeurčená. Přijmeme-li totiž od experta hodnoty $P(H \mid E)$, $P(H \mid \text{not}E)$ a $P(E)$, pak již nemůžeme přijmout expertem určenou hodnotu $P(H)$, nýbrž hodnotu $P_C(H)$, která vychází podle dané lineární funkce. Takový systém pravděpodobnostních měr se stává nekonzistentním.

K odstranění tohoto sporu, tj. k dodržení konzistence systému subjektivních pravděpodobností, se přijímá řešení, které znamená **odklon od konvenčního bayesovského přístupu.** Systém PROSPECTOR používá po částech lineární aproximace upravené funkce a toto řešení bývá použito i u dalších systémů, využívajících pravděpodobnostního principu.

Studenti budou v rámci konzultací seznámeni s prázdným pravděpodobnostním expertním systémem FEL-EXPERT [5] (zájemci si mohou tento programový systém stáhnout z webovské stránky http://homen.vsb.cz/~pok40/).



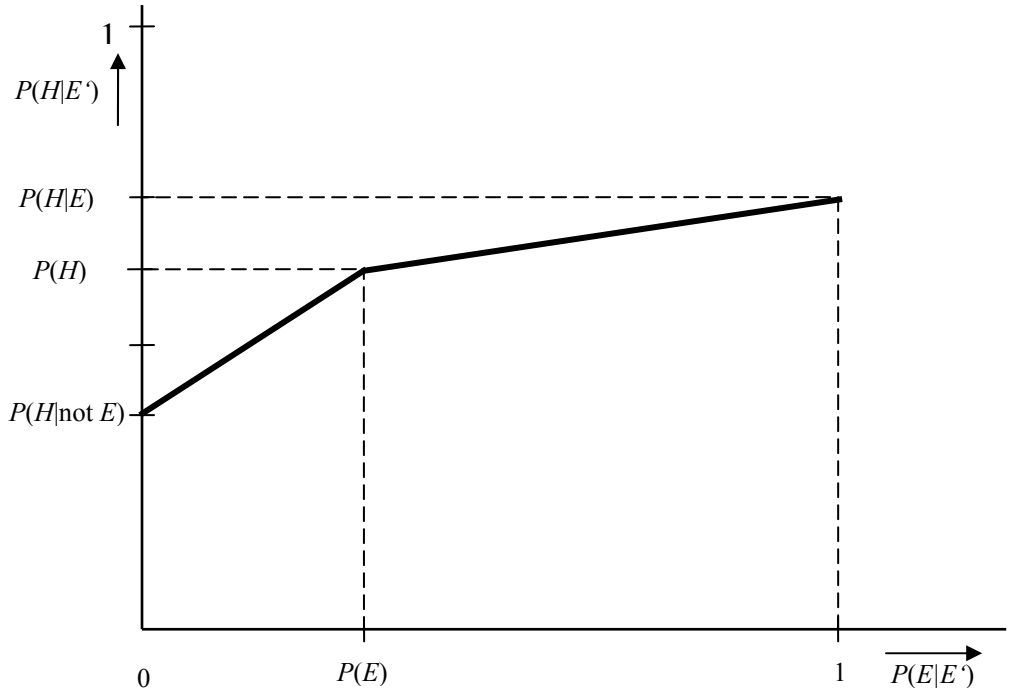
Obr.4.2

V našem případě, uvedeném na Obr.4.3, se pro výpočet $P(H \mid E')$ používá vztahů

$$P(H \mid E') = P(H) + \frac{P(H \mid E) - P(H)}{1 - P(E)} \cdot (P(E \mid E') - P(E))$$
$$\text{pro } P(H \mid E) \geq P(H) \geq P(H \mid \text{not } E)$$

$$P(H \mid E') = P(H \mid \text{not } E) + \frac{P(H) - P(H \mid \text{not } E)}{P(E)} \cdot P(E \mid E')$$
$$\text{pro } P(H \mid E) \leq P(H) \leq P(H \mid \text{not } E)$$

Nyní je třeba zabývat se opět otázkou, jak kombinovat vliv váhy více předpokladů E_1, E_2, ..., E_n na stejný závěr H.



Obr.4.3

Nejprve uvažujme situaci, kdy předpoklady nabývají pouze logických hodnot „pravda“ nebo „nepravda“. Zde by bylo možno použít složený logický předpoklad $E = E_1 \,\&\, E_2 \,\&\, ... \,\&\, E_n$ a ve vztazích (4.20) a (4.21) použít pravděpodobnostní šance

$$O = \frac{P(E_1 \,\&\, E_2 \,\&\, ... \,\&\, E_n \mid H)}{P(E_1 \,\&\, E_2 \,\&\, ... \,\&\, E_n \mid \text{not } H)}$$

Podmínka statistické nezávislosti tvrzení E, který je pro prospektorovský model předpokládán, nám umožňuje aktualizovat pravděpodobnostní šance podle vztahů

$$O = (H \mid E_1, E_2, ..., E_n) = L_1 \,.\, L_2 .... L_n \,.\, O(H) \tag{4.25}$$

$$O = (H \mid \text{not } E_1, \text{not } E_2, ..., \text{not } E_n) = \hat{L}_1 \,.\, \hat{L}_2 .... \hat{L}_n \,.\, O(H) \tag{4.26}$$

Obecně však předpoklad statistické nezávislosti splněn není. Abychom se mu alespoň přiblížili, doporučuje se dodržovat zásadu, aby počet pravidel se stejnou levou stranou byl malý a také počat pravidel se stejnou pravou stranou (hypotézou) byl opět malý.

Předpokládejme nyní, že uživatel přiřazuje evidencím E na základě svého pozorování určitou váhu, označovanou $E'_1, \; E'_2, \; ..., \; E'_n$. Pak lze definovat tzv. **efektivní váhu pravidla**

$$L'_i = \frac{O(H \mid E'_i)}{O(H)}$$

a výslednou pravděpodobnostní šanci O dostaneme (opět za předpokladu statistické nezávislosti evidencí) podle vztahu analogického k (4.25) a (4.26)

$$O(H \mid E'_1, E'_2, ...E'_n) = L'_1 \,.\, L'_1 \,...L'_n \,.\, O(H)$$

Odtud získáme podle vztahu

$$P = \frac{O}{O+1}$$

pravděpodobnost hypotézy H.

Závěrem shrneme:

expert tedy zadává ke každému pravidlu buď dvojici měr L a $\hat{L}$ nebo dvojici subjektivních pravděpodobností $P\,(H \mid E)$ a $P\,(H \mid \text{not}E)$ a dále pak apriorní pravděpodobnosti $P\,(E)$ a $P\,(H)$. Uživatel (nebo báze dat) zadá hodnoty $P\,(E \mid E')$. Tím je pravděpodobnost $P\,(H \mid E'_1, E'_2, ..., E'_n)$ plně určena.

Pro výpočet pravděpodobnostního ohodnocení logické kombinace dílčích výroku se v prospektorovském modelu používá vztahů, převzatých z teorie fuzzy množin (viz kap. 5).

$$P(E_1 \,\&\, E_2) = \min\{P(E_1), P(E_2)\}$$
$$P(E_1 \vee E_2) = \max\{P(E_1), P(E_2)\}$$
$$P(\text{not } E) = 1 - P(E)$$

4.2.1 Expertní systém typu PROSPECTOR

Použijme opět fragment pravidel diagnostického systému poruch automobilového motoru jako v příkladu inference MYCIN. Ohodnocení je pozměněno, aby odpovídalo modelu prospektorovskému.

IF $\langle E_1$: Klepou ventily při přidání plynu$\rangle$

THEN $\langle H$: Špatně seřízený předstih$\rangle$

WITH $\langle P(H \mid E_1) = 0{,}85; P(H \mid \text{not } E_1) = 0{,}05\rangle$

IF $\langle E_2$: Motor střílí do výfuku$\rangle$

THEN $\langle H$: Špatně seřízený předstih$\rangle$

WITH $\langle P(H \mid E_2) = 0{,}95; P(H \mid \text{not } E_2) = 0{,}3\rangle$

IF $\langle E_3$: Malá spotřeba$\rangle$

THEN $\langle H$: Špatně seřízený předstih$\rangle$

WITH $\langle P(H \mid E_3) = 0{,}001; P(H \mid \text{not } E_3) = 0{,}7\rangle$

IF $\langle E_4$: Malá akcelerace$\rangle$

THEN $\langle H$: Špatně seřízený předstih$\rangle$

WITH $\langle P(H \mid E_4) = 0{,}9; P(H \mid \text{not } E_4) = 0{,}1\rangle$

Předpokládejme, že jsme od experta získali následující apriorní pravděpodobnosti:

$P(E_1) = 0{,}1$; $P(E_2) = 0{,}05$; $P(E_3) = 0{,}6$; $P(E_4) = 0{,}4$ a $P(H) = 0{,}35$.

Uživatel postupně zadává odpovědi:

a) $P(E \mid E'_1) = 0{,}8$, pak

$$P(H \mid E'_1) = P(H) + \frac{P(H \mid E_1) - P(H)}{1 - P(E_1)} \cdot (P(E_1 \mid E'_1) - P(E_1) = 0{,}739$$

$$L'_1 = \frac{O(H \mid E'_1)}{O(H)} = \frac{\dfrac{P(H \mid E'_1)}{1 - P(H \mid E'_1)}}{\dfrac{P(H)}{1 - P(H)}} = 5{,}258$$

přičemž

$$O(H) = 0{,}5385; O(H \mid E'_1) = 2{,}831$$

b) $P(E \mid E'_2) = 0{,}01$, pak

$$P(H \mid E'_2) = P(H \mid \text{not } E_2) + \frac{P(H) - P(H \mid \text{not } E_2)}{P(E_2)} \cdot P(E_2 \mid E'_2) = 0{,}301$$

$$L'_2 = \frac{O(H \mid E'_2)}{O(H)} = 0{,}8$$

a tedy

$$O(H \mid E'_1, E'_2) = L'_1 \,.\, L'_2 \,.\, O(H) = L'_2 \,.\, O(H \mid E'_1) = 0{,}8 \,.\, 2{,}831 = 2{,}265$$

Odtud

$$P(H \mid E'_1, E'_2) = 0{,}694$$

c) $P(E \mid E'_3) = 0071$, pak analogicky jako v předchozích případech vypočítáme

$$P(H \mid E'_3) = 0{,}071$$

$$L'_3 = 0{,}142$$

$$O(H \mid E'_1, E'_2, E'_3) = L'_1, L'_2, L'_3 \,.\, O(H) = L'_3 \,.O(H \mid E'_1, E'_2) =$$
$$= 0{,}142 \,.\, 2{,}265 = 0{,}322$$

$$P(H \mid E'_1, E'_2, E'_3) = 0{,}244$$

d) po odpovědi na poslední otázku uvede $P(E \mid E'_4) = 0{,}8$, pak

$$P(H \mid E'_4) = 0{,}717$$

$$L'_4 = 4{,}705$$

$$O(H \mid E'_1, E'_2, E'_3, E'_4) = L'_4 \,.\, O(H \mid E'_1, E'_2, E'_3) = 4{,}705 \,.\, 0{,}224 = 1{,}148$$

$$P(H \mid E'_1, E'_2, E'_3, E'_4) = 0{,}534$$

Model pro práci s neurčitostí typu PPROSPECTOR - shrnutí

Model PROSPECTOR používá modifikovaného Bayesova vztahu pro dokazování platnosti hypotéz. Používá produkčních pravidel, která mohou nebo nemusí být provázena neurčitostí. Pro ohodnocení pravidel používá tento systém kritéria naděje nebo pravděpodobnostní šance. Pracuje s pojmem míry nezbytnosti. Pro ohodnocení konkrétní situace používá subjektivní ohodnocení. Jeho řídicí mechanizmus opět ohodnocuje jednotlivé hypotézy a stanoví tu, která nejlépe koresponduje s daty konkrétního případu. Použití modelu PROSPECTOR je opět ilustrováno příkladem jednoduchého diagnostického expertního systému.

Kontrolní otázky:

1. Jak se liší přístupy systémů MYCIN a PROSPECTOR v metodě řešení problémů?
2. V čem spočívá modifikace klasického Bayesova vztahu dokazování platnosti hypotéz, kterou využívá systém PROSPECTOR?

Otázka k zamyšlení:

1. Mají význam pravděpodobnostní expertní systémy, používající pravidla bez neurčitosti? Kdy je možné jich použít?

Úkoly k zamyšlení:

Pokuste se formulovat úlohu řešitelnou přístupem pravděpodobnostního expertního systému, která umožňuje použití pravidel bez nejistoty.

Korespondenční úkoly:

Navrhněte a vypracujte návrh expertního systému na principech modelu MYCIN. K realizaci použijte prostředí FEL-EXPERT, s nímž budete seznámeni při konzultacích (zájemci si mohou tento programový systém stáhnout z webovské stránky http://homen.vsb.cz/~pok40/).

.

Shrnutí obsahu kapitoly

Expertní systém MYCIN je jedním z prvních úspěšných expertních systémů. Využívá reprezentace znalostí formou produkčních pravidel. Neurčitost předpokladů je vyjádřena pomocí míry důvěry a míry nedůvěry v platnost pravidla. Tyto míry jsou spojeny do činitele jistoty. Neurčitost dat je stanovena na základě pozorování odhadem míry jistoty v platnost splnění předpokladů. Řídicí ad-hoc mechanizmus provede ohodnocení pravidel a stanoví nejpravděpodobnější hypotézu jako závěr. Model PROSPECTOR používá modifikovaného Bayesova vztahu pro dokazování platnosti hypotéz. Používá produkčních pravidel, která mohou nebo nemusí být provázena neurčitostí. Pro ohodnocení pravidel používá tento systém kritéria naděje nebo pravděpodobnostní šance. Pracuje s pojmem míry nezbytnosti. Pro ohodnocení konkrétní situace používá subjektivní ohodnocení. Jeho řídicí mechanizmus opět ohodnocuje jednotlivé hypotézy a stanoví tu, která nejlépe koresponduje s daty konkrétního případu.

Doporučená literatura

[1] Shortliffe,E.H.: Compurer.Based Medical Consultations, Elsevier, New York, 1976
[2] van Melle,W.: A Domain-Independent System That Aids in Constructing Knowledge-Based Consultation Programs, Memo HPP-80-22, Stanford University, Stanford, 1980
[3] Duda,R.O. a kol.: Development of PROSPECTOR Consultation System for Mineral Exploration, Technical Report, AI Center, SRI International, Menlo Park, 1976
[4] Kotek,Z.a kol.: Kybernetika, SNTL Praha, 1987
[5] Mařík,V,. Zdráhal,Z.: Expertní systémy, UISK, Praha, 1987

5. FUZZY ORIENTOVANÉ EXPERTNÍ SYSTÉMY

V této kapitole se dozvíte:

- Historii a důvody vzniku fuzzy množin
- Jaké jsou principy fuzzy množinové teorie ve srovnání s teorií množin klasických
- Základy fuzzy množinových operací, fuzzy relací a fuzzy logiky
- Jak je fuzzy logika využita v metodě přibližného usuzování
- Něco o principech fuzzy modelování složitých soustav včetně příkladu
- Způsob konstrukce fuzzy expertních systémů včetně praktického příkladu

Po jejím prostudování byste měli být schopni:

- Vysvětlit principy fuzzy množin a jejich použití pro reprezentaci vágnosti pojmů
- Objasnit základy fuzzy logiky a jejího použití v přibližném usuzování
- Navrhnout a realizovat fuzzy model složité soustavy
- Popsat, navrhnout a realizovat jednoduchý fuzzy orientovaný expertní systém

Klíčová slova této kapitoly:

Fuzzy množina, funkce příslušnosti, fuzzy množinové operace, operace t-normy, operace s-normy, fuzzy relace, fuzzy logika, fuzzy přibližné usuzování, fuzzy model, fuzzy expertní systém, defuzifikace

Doba potřebná ke studiu: 6 hodin

iem

Poslední kapitola modulu je kapitolou nejrozsáhlejší. Je věnována prakticky velmi rozšířené metodě tvorby expertních systémů, která využívá teorie fuzzy množinové matematiky a fuzzy logiky. Tyto přístupy jsou totiž nejen základem tvorby fuzzy orientovaných expertních systémů, mají velký význam i v širším kontextu jazykového fuzzy modelování a simulace chování složitých soustav. Seznámíte se postupně se všemi teoretickými i praktickými aspekty použití fuzzy množin ve fuzzy logice a fuzzy přibližném usuzování. Naučíte se také navrhovat a realizovat fuzzy expertní systém, což mů

Při konzultacích budete seznámeni s vývojovým systémem LMPS, v němž můžete své projekty realizovat. Využívejte doplňkové doporučené literatury a rady tutora.

5.1 Paradigma fuzzy množin

Jak již bylo řečeno, základní myšlenka expertního systému spočívá ve využití v počítači uložených kvalitních znalostí experta k řešení úloh z jeho oboru v případě, že expert není v daném okamžiku k dispozici (zkušený a vynikající lékař, primář špičkového pracoviště, odešel do důchodu). Řekli jsme, že kvalita expertního systému je dána především kvalitou znalostí a efektivitou jejich počítačové reprezentace.

Ať již jsou znalosti uloženy v bázi znalostí jakýmkoliv způsobem, vždy jsou získány jejich „extrakcí“ od experta. Nejpřirozenějším a nejoperativnějším vyjádřením takových znalostí je přirozený jazyk ať už v ústní nebo písemné formě. Dále bylo praxí prověřeno, že snad nejefektivnější formou vyjádření lidských znalostí (např. o chování nějakého objektu) jsou produkční pravidla typu JESTLIŽE-PAK. Ukazuje se, že souborem takových pravidel lze popsat chování libovolně složité soustavy.

Produkční pravidla, vyslovená v přirozené řeči, jsou složena ze slov přirozeného jazyka. Základní vlastností slovních pojmů je jejich pojmová neurčitost **- vágnost.** Jazykový pojem „vysoký strom“ je vágní, v daném kontextu (praktická výška stromů) jej však člověk s určitou životní zkušeností (zde ani nemusíme hledat experta na výšku stromů) dovede dobře vyhodnotit a efektivně použít. Pokud požadujeme po expertním systému, aby podobně dobře operoval se znalostmi a jsou-li tyto znalostí formulovány produkčními pravidly s využitím jazykových popisů, stojíme nutně před problémem nalezení vhodného nástroje pro počítačovou formalizaci jejich pojmové neurčitosti, tedy vágnosti.

V praxi nejvíce rozšířeným způsobem formalizace vágních pojmů jako formalizace neurčitosti slov přirozeného jazyka se staly tzv. **fuzzy množiny.**

Podívejme se nyní, v čem spočívá základní idea, která vedla ke vzniku teorie fuzzy množin. Zakladatelem fuzzy množinové matematiky je americký profesor kalifornské univerzity v Berkeley Lotfi Asker Zadeh (1965) [1].

Moderní matematika je budována na teorii množin. Základem jejich použití je rozhodování, z jakých prvků se množina skládá. Jde o rozhodování, zda určitý prvek (nebo objekt) do určité množiny prvků nebo objektů (vyznačující se určitou vlastností) náleží nebo nenáleží. Z tohoto hlediska je třeba každému prvku (objektu) přiřadit číselný parametr - tzv. **stupeň jeho náležení** do určité množiny, který logicky nabývá hodnot 1 (prvek do množiny bez jakýchkoliv pochyb náleží) nebo 0 (prvek do množiny bez jakýchkoliv pochyb nenáleží).

Kamenem úrazu se však stává právě ono rozhodnutí. Přiřazení stupně náležení určitého prvku do určité množiny je problém, ležící většinou mimo matematiku a je obvykle vůbec těžko rozhodnutelný.

Pro ilustraci uveďme následující příklad [2]. Uvažujme množinu všech lidí, kteří kdy na zemi žili. Bude to zřejmě množina spočetná, budou tam patřit všichni žijící lidé a jejich zemřelí předkové. Ale pozor – lidský druh se vyvíjel a musíme rozhodnout, které jeho historické předky už lze považovat za lidi a do množiny tedy patřit budou a které předky ještě za lidi považovat nebudeme a do množiny je nezahrneme! Zcela jistě bude do této množiny patřit zakladatel této vývojové teorie Charles Darwin. Bude tam ale patřit o několik stovek generací starší člen vývojové řady – opičák pracovně nazvaný Charlie? O tom se budou velmi obtížně dohadovat paleontologové, antropologové atd., natož matematici. *Nelze popřít skutečnost, že opičák Charlie má s lidským druhem mnohé rysy společné a proto tak trochu do množiny lidí patří.* Jak vidíme, přirozený jazyk si tímto problémem poradí – **použije vágního pojmu „tak trochu“**. Poradí si s ním však klasická množinová teorie, která zná pouze dvě možnosti – buď absolutně náleží nebo absolutně nenáleží, avšak nedovede formalizovat náleží „tak trochu“!

Přesuňme nyní svoji pozornost z oblasti množinové matematiky do oblasti logiky. Existuje zde určitá spojitost – na hodnotu stupně náležení se můžeme dívat jako na pravdivostní hodnotu logického výroku „patří do množiny“. V oblasti logiky je situace poněkud jiná – nekonvenční logiky (např. logika Lukasiewiczova [3]) používají obor pravdivostních hodnot reálných čísel z intervalu $\langle 0, 1 \rangle$. Pravdivostní hodnotu výroku

A: „Charles Darwin patří do množiny lidí“

pak jistě vyjádříme hodnotou P(*A*) = 1, pro vyjádření pravdivostní hodnoty výroku

B: „Opičák Charlie patři do množiny lidí“

máme možnost použít hodnotu např. P(*B*) = 0,1. Tento zdánlivě jednoduchý krok otevřel cestu nesmírným možnostem.

Přenesme se zpět do světa teorie množin. Ukázali jsme souvislost mezi problémem kvantifikace příslušení prvku do množiny a kvantifikace pravdivostní hodnoty logického výroku. Po bitvě bývá každý generálem – profesor Zadeh provedl dnes evidentně jasnou úvahu, která vedla k zavedení pojmu **částečného náležení prvku do množiny**, ekvivalentního pojmu částečné pravdivosti logického výroku. Stupeň příslušnosti prvku do – nové, tzv. fuzzy – množiny bude tedy definován jako reálné číslo z intervalu ⟨0, 1⟩. Toto řešení ponechává vyjádření absolutní příslušnosti (1) a absolutní nepříslušnosti (0), umožňuje však formalizaci příslušnosti částečné, kvantifikované stupněm z intervalu (0, 1). Jde tedy z jistého pohledu o zobecnění klasických množin na množiny nové – **fuzzy množiny** (fuzzy - anglický výraz ekvivalentnímu českému nejasný, nezřetelný, ne zcela vymezený, mlhavý - se do češtiny nepřekládá). V odborné terminologii pak hovoříme o obyčejných množinách a fuzzy množinách.

Vraťme se nyní k problematice formalizace jazykových výrazů ve smyslu reprezentace jejich pojmové neurčitosti – vágnosti. Fuzzy množiny, jak uvidíme dále, mohou sloužit jako excelentní prostředek pro formalizaci jazykových výrazů jako „malý“, „střední“, „velmi starý“, „mladý“, „asi nulový“ apod. V následující kapitole se seznámíme se základy fuzzy množinové teorie a fuzzy přístupy používanými v expertních systémech.

Paradigma fuzzy množin - shrnutí

Kapitola vysvětluje pragmatické důvody, které vedly ke zobecnění teorie klasických množin. Obtíže s přiřazováním jednoznačného stupně příslušnosti nebo jednoznačné nepříslušnosti prvku do množiny v případech reálného světa vedly z zavedení pojmu příslušnosti částečné a ke vzniku tzv. fuzzy (mlhavých) množin. Vysvětlení jejich praktického významu je dokumentováno na příbuzných příkladech problému stanovení pravdivosti výroků v klasické logice. Je zaveden pojem neurčitosti pojmů přirozeného jazyka – vágnosti.

Kontrolní otázky:

1. Jaký je vztah klasických a fuzzy množin?
2. V čem spočívá neurčitost jazykových výrazů a čím je v procesu lidského chápání omezována?

Otázka k zamyšlení:

1. Které třídy prvků lze formalizovat klasickými množinami?

Úkoly k zamyšlení:

Promyslete několik výroků a přiřaďte jim pravdivostní hodnoty podle vlastního uvážení. Srovnejte je s hodnotami, kterými je vybaví váš kamarád!

5.2 Základy teorie fuzzy množin

Stupeň příslušnosti (náležení) prvku univerza U do klasické množiny A [4], [5] je dán funkcí γ_A

$$\gamma_A : U \to [0,1]. \tag{5.1}$$

Její velikost může nabývat dvou hodnot:

- stupeň příslušnosti 0 – prvek do množiny A (plně) nenáleží
- stupeň příslušnosti 1 – prvek do množiny A (plně) náleží.

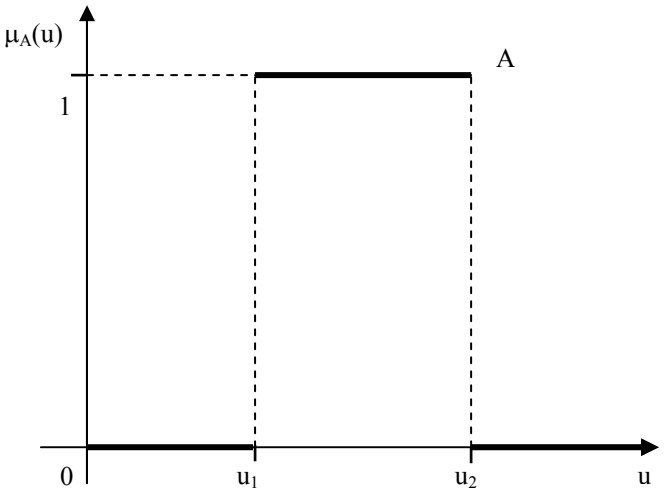
Jak jsme již dříve konstatovali, v převážné většině praktických případů lze jen obtížně tvrdit, že určitý prvek do určité množiny náleží či nenáleží. I když prvek nese dominantní znaky vlastností prvků určité množiny A, může ve více méně menší míře vykazovat také znaky vlastností množiny B. Rozhodnutí o přiřazení prvku do množiny A je pak nejednoznačné.
Situaci může efektivně řešit přístup, v němž kromě pojmu absolutního náležení či nenáležení prvku do určité množiny zavedeme pojem částečného náležení prvku do množiny. Jde zřejmě o zobecnění pojmu stupně příslušnosti (1), kdy rozšíříme definiční obor jeho hodnot ze dvou diskrétních (0, 1) na uzavřený interval <0, 1>

$$\mu_F : U \to \langle 0,1 \rangle . \tag{5.2}$$

Množiny, které umožňují definovat velikost stupně náležení prvků podle (5.2) se nazývají **fuzzy množiny** [Zadeh]. Fuzzy množina F je definována jako přiřazení, které každému prvku u univerza U přiřazuje hodnotu funkce jeho příslušnosti do fuzzy množiny F rovnou $\mu_F(u)$

$$F = \{(\mu_F(u))/u \in U\}. \tag{5.3}$$

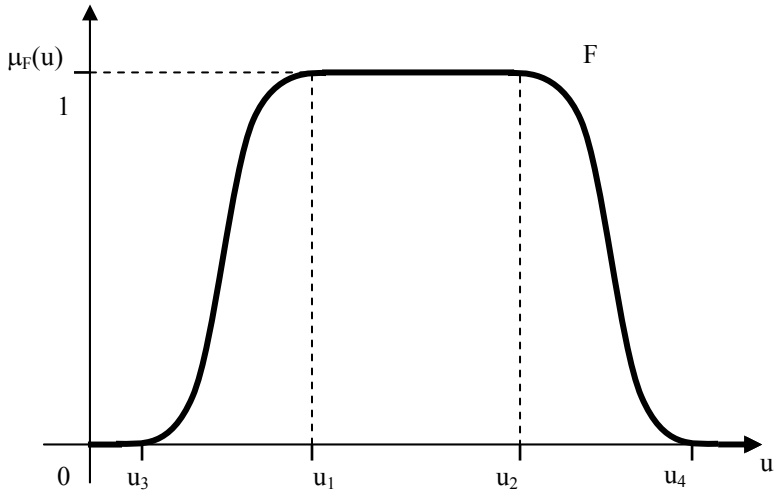
V praxi je fuzzy množina F prvků u ztotožněna s její funkci příslušnosti $\mu_F(u)$. Při popisu grafického průběhu funkce příslušnosti $\mu_F(u)$ vyjdeme nejprve z průběhu funkce příslušnosti $\mu_A(u)$ klasické množiny A, nakresleného na Obr.5.1.



Obr.5.1

Prvky $u \in U$ z uzavřeného intervalu $\langle u_1, u_2 \rangle$ do množiny A zcela jistě patří, ostatní prvky univerza do množiny A zcela jistě nepatří.

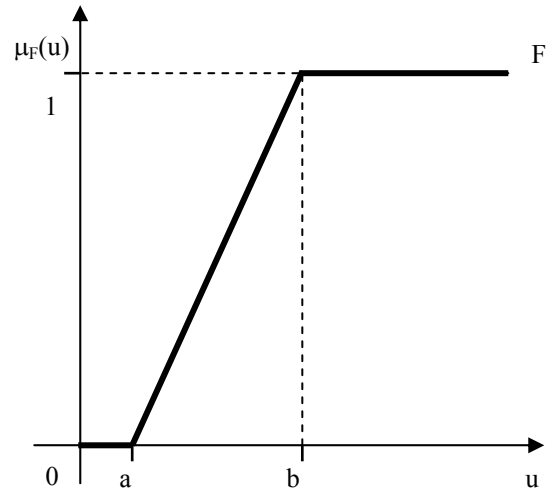
Na Obr.5.2 je nakreslena funkce příslušnosti fuzzy množiny F, definované opět na univerzu U. Prvky $\langle u_1, u_2 \rangle$ opět do fuzzy množiny F zcela jistě patří, prvky u z intervalu $(-\infty, u_3\rangle$ a prvky u z intervalu $\langle u_4, \infty)$ do fuzzy množiny F zcela jistě nepatří.



Obr.5.2

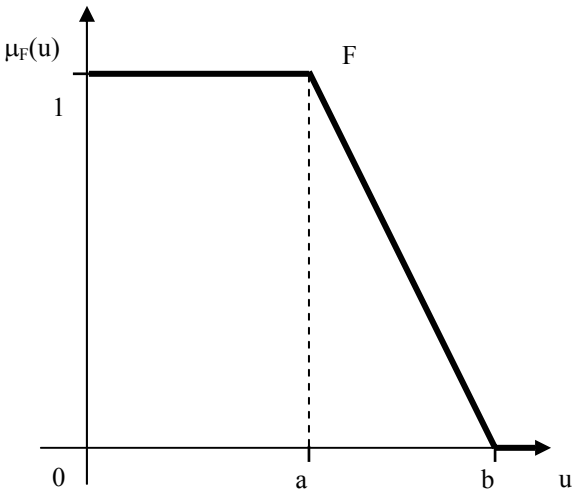
Prvkům u z intervalů (u_3, u_1) a (u_2, u_4) je funkcí $\mu_F(u)$ přiřazena hodnota jejich příslušnosti do fuzzy množiny F reálným číslem z intervalu (0, 1) a vyjadřuje tedy jejich **příslušení částečné**.

Praktické použití fuzzy množin vyžaduje analytické vyjádření funkce $\mu_F(u)$. V praxi používáme nejčastěji jejich aproximaci lomenými přímkami. Příklady takových aproximací a jejich analytické odpovídající aproximaci jsou uvedeny na Obr.5.3 až Obr.5.6. Funkce příslušnosti jsou parametrizovány jejich čtyřmi body zlomu, tedy hodnotami $[a, b, c, d]$.



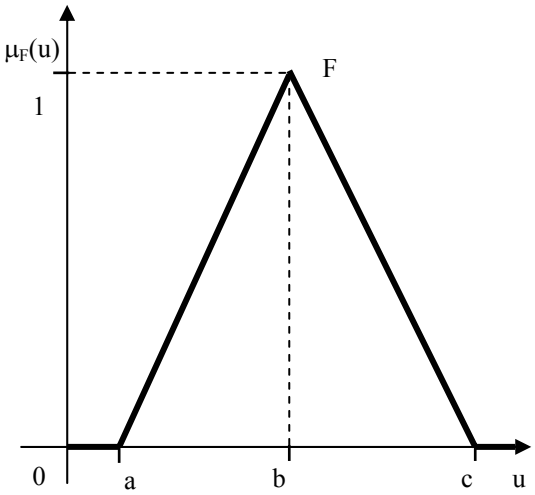
$$\Gamma(u,a,b)=\begin{cases}0 & u<a\\ (u-a)/(b-a) & a\le u\le b\\ 1 & u>b\end{cases}$$

Obr.5.3



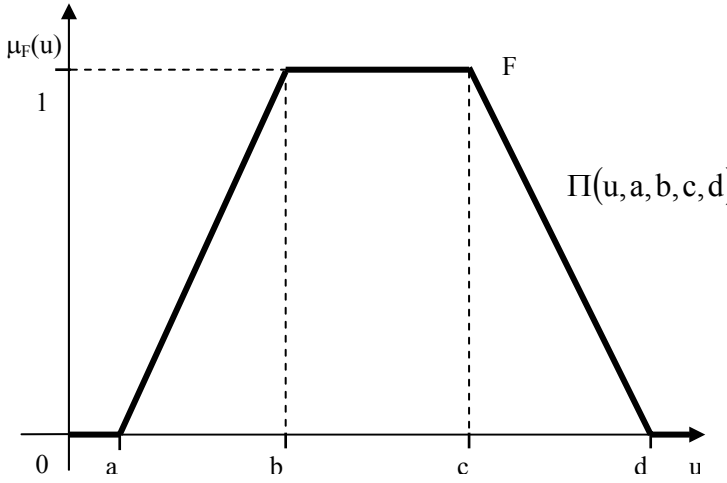
$$L(u,a,b)=\begin{cases}1 & u<a\\ (b-u)/(b-a) & a\le u\le b\\ 0 & u>b\end{cases}$$

Obr.5.4



$$\Lambda(u,a,b,c)=\begin{cases}0 & u<a\\ (u-a)/(b-a) & a\le u\le b\\ (c-u)/(c-b) & b\le u\le c\\ 0 & u>c\end{cases}$$

Obr.5.5



$$\Pi(u,a,b,c,d)=\begin{cases}0 & u<a\\ (u-a)/(b-a) & a\le u\le b\\ 1 & b\le u\le c\\ (d-u)/(c-d) & c\le u\le d\\ 0 & u>d\end{cases}$$

Obr.5.6

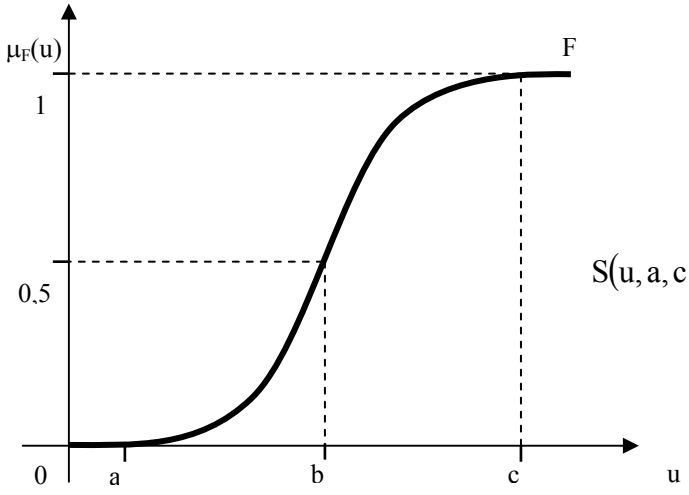
Případ fuzzy množiny, jejíž funkce příslušnosti má trojúhelníkový tvar (Obr.5.5), má zvlášní důležitost. Taková fuzzy množina totiž formalizuje ne zcela přesné, tzv. fuzzy číslo, v daném případě reprezentované jazykovým výrazem „asi b“. Fuzzy čísla (vedle čísel obyčejných, ostrých – zde „přesně b“) mají velký praktický význam, jak uvidíme dále.

Fuzzy množinu *A* definovanou na univerzu *X* nazýváme normální, pokud má výšku 1, neboli

$\exists\, x \in X,\ \mu_A(x) = 1.$

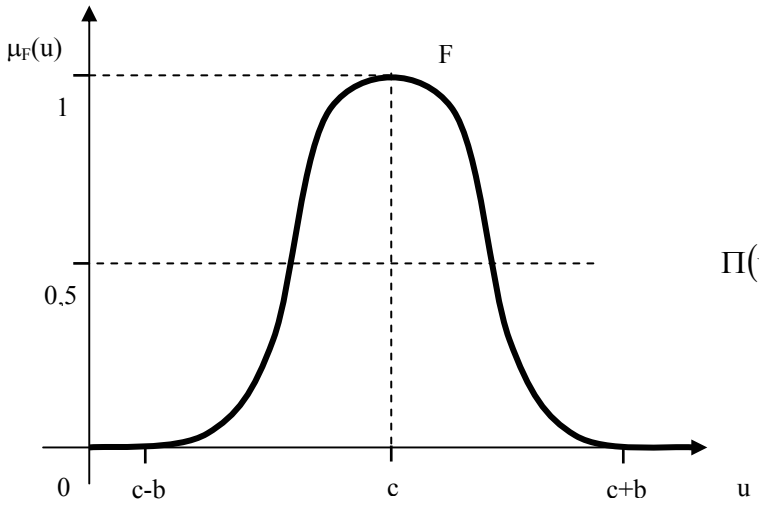
Uvědomíme si, že ostré číslo můžeme rovněž reprezentovat fuzzy množinou (normální fuzzy množina definovaná na jediném prvku univerza). Takovou fuzzy množinu pak nazýváme singleton. Singletonem je fuzzy množina, odpovídající obyčejnému ostrému číslu.

Aproximace funkce příslušnosti fuzzy množiny spojitou funkcí je uvedena na Ob.5.7 a Obr.5.8. Křivky jsou parametrizovány dvěma hodnotami – [*a,c*] resp. [*b,c*].



$$S(u,a,c)=\begin{cases}0 & u\le a\\ 2(u-a)^2/(c-a)^2 & a<u\le b\\ 1-2(u-c)^2/(c-a)^2 & b<u\le c\\ 1 & u>c\end{cases}$$

Obr.5.7



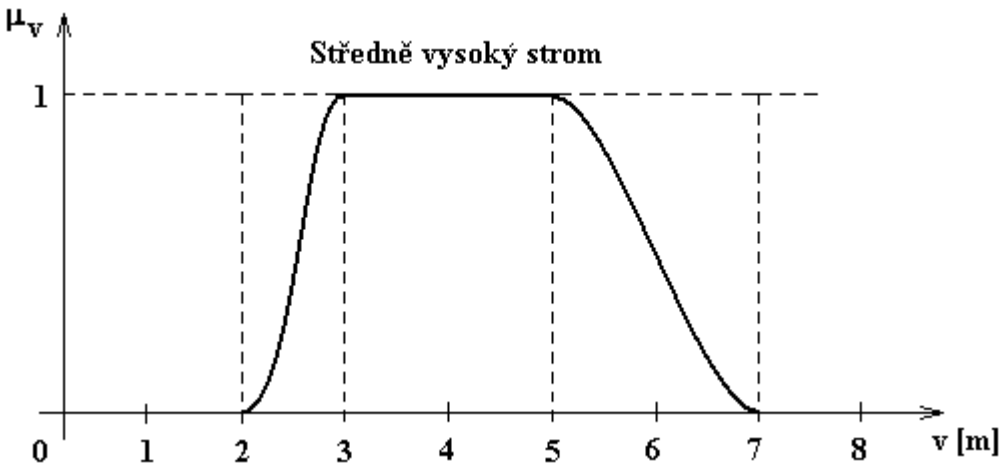
$$\Pi(u,b,c)=\begin{cases} S(u,c-b,c) & u \le c \\ 1-S(u,c,c+b) & u \ge b \end{cases}$$

Obr.5.8

5.2.1 Formalizace vágních pojmů

Pokusme se popsat pomocí fuzzy množiny vágní pojem "středně vysoký strom". Každé výšce, která připadá v úvahu, přiřadíme číslo *<0,1>* vyjadřující stupeň našeho přesvědčení, že takový strom je "středně vysoký". Tento stupeň vyplývá z toho, jak rozumíme pojmu "výška stromu", jak hluboká je v tomto ohledu naše zkušenost, jak dalece jsme v této oblasti experty. Přiřazení takových stupňů (stupňů příslušnosti) závisí tedy na subjektu a také na kontextu. Proto přiřazení funkce příslušnosti prvkům fuzzy množiny je z principu subjektivní a odráží obecně koncept, z něhož je problém posuzován.

Pokusme se tedy definovat fuzzy množinu, formalizující vágní pojem "středně vysoký strom". Jedna z možností znázornění takové fuzzy množiny je grafický způsob, kdy na vodorovnou osu budeme vynášet výšky, na svislou osu odpovídající stupně příslušnosti. Možná definice je zobrazena na Obr.5.9.

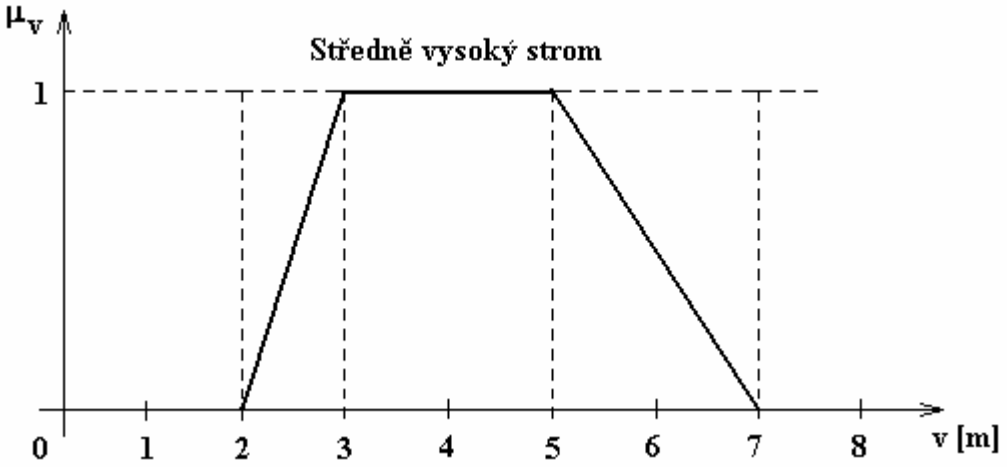


Obr.5.9

Takto jsou definovány jako "zcela určitě středně vysoké" stromy s výškou mezi 3m a 5m, (stupeň příslušnosti 1); naproti tomu mezi "středně vysoké" nejsou zařazeny zcela určitě stromy nižší než 2m a vyšší než 7m (stupeň příslušnosti 0). Čtenář si zcela určitě vytvoří podobné fuzzy množiny sám ("staré auto", "mladý člověk", "hodně peněz" a pod.).

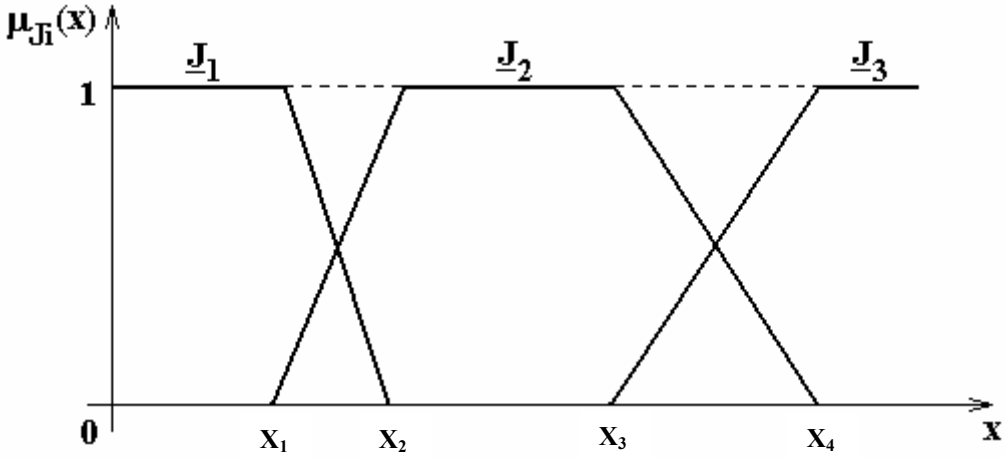
Spojitá křivka zvonového tvaru, představující průběh velikosti stupně příslušnosti v závislosti na velikosti prvku univerza, je nazývána **fuzzy charakteristikou** formalizovaného pojmu, která je ztotožňována s příslušnou fuzzy množinou „středně vysoký strom“.

Taková křivka bývá často parametrizována čtyřmi body zlomu (*a,b,c,d*) – v našem případě (2, 3, 5, 7) a aproximována lomenými přímkovými úseky, jak je uvedeno na Obr.5.10.



Obr.5.10

Jako příklad formalizace více jazykových hodnot jedné jazykové proměnné můžeme dále uvést již zmíněnou jazykovou proměnnou "výška stromu", která může nabývat tří jazykových hodnot "nízký" J_1, "středně vysoký" J_2 a "vysoký" J_3. Tyto tři jazykové hodnoty jsou reprezentovány třemi fuzzy množinami, a to typy Z, Π a S, podle Obr.5.11.



Obr.5.11

Každá jazyková hodnota J_i je pak určena uspořádanou čtveřicí svých parametrů. Uspořádaná čtveřice představuje body zlomu aproximační přímky a čtyři hodnoty tvoří v takovém případě čtyři parametry fuzzy množiny.

Základy teorie fuzzy množin - shrnutí

Základy teorie fuzzy množin seznamují s principy formalizace fuzzy množiny její funkcí příslušnosti a jejími praktickými aproximacemi. Uvádí příklad použití fuzzy množin při formalizaci jazykového pojmu, prozatím intuitivně zavádí termín jazykové proměnné a jejích jazykových hodnot.

Kontrolní otázky:

1. Jak je definována funkce příslušnosti prvku k obyčejné a fuzzy množině?
2. Jaké jsou způsoby aproximace funkcí příslušnosti?
3. Co je to fuzzy číslo?

Otázka k zamyšlení:

1. Lze ostré (obyčejné) číslo považovat za fuzzy množinu?

Úkoly k zamyšlení:

Pomocí fuzzy množin formalizujte libovolnou jazykovou proměnnou se čtyřmi jazykovými hodnotami, z nichž jedna je fuzzy číslo.

5.3 Fuzzy množinové operace

Uveďme nejprve další možné formy zápisu funkce příslušnosti fiuzzy množiny *F*. Je-li fuzzy množina *F* definována na diskrétním univerzu s *n*- prvky, lze ji určit výčtem dvojic $\mu_F(u)/u$ ve tvaru zápisu

$$F = \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, ...\mu_F(u_n)/u_n\}$$

V tomto zápisu se někdy používá znaménko + a konečnou množinu *F* lze zapsat ve tvaru

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + ... + \mu_F(u_n)/u_n$$

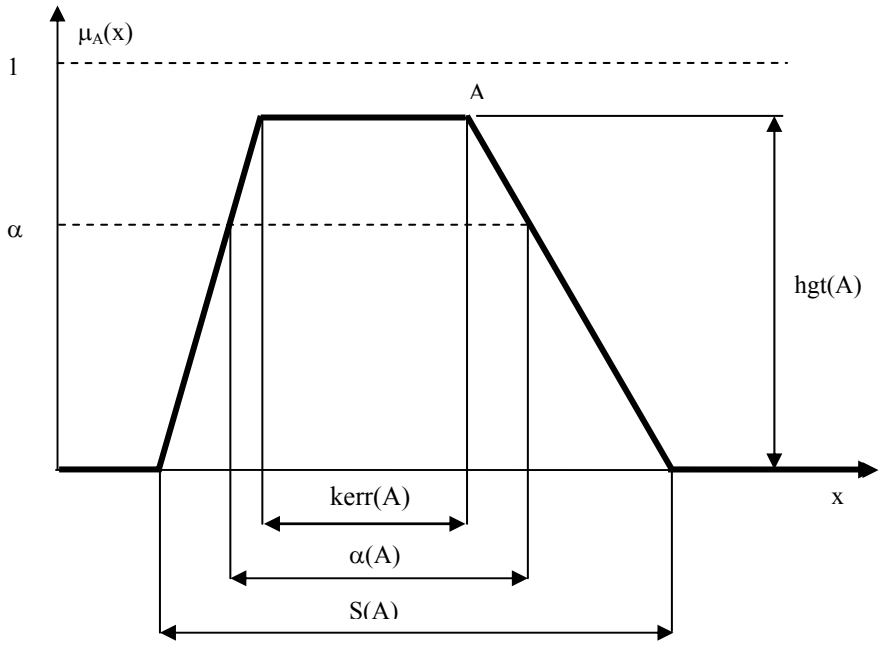
Fuzzy množinu *F*, definovanou na diskrétním konečném nebo spočetném univerzu *U* můžeme formalizovat zápisem

$$F = \sum_{u \in U} \mu_F(u)/u$$

v případě univerza spojitého nebo nespočetného pak formálně použijeme symbol integrálu

$$\int_U \mu_F(u)/u$$

Pro kvantitativní vyjádření tvaru funkce příslušnosti fuzzy množiny A použijeme následující parametry, jejichž význam plyne z Obr.5.12:



Obr.5.12

nosič fuzzy množiny A

$$S(A) = \{x / \mu_A(x) \rangle 0\}$$

jádro fuzzy množiny A

$$\ker r(A) = \{x \in X / \mu_A(x) = 1\}$$

výšku fuzzy množiny A

$$hgt(A) = \sup_{x \in X}(\mu_A(x))$$

a konečně α- řez fuzzy množiny A

$$\alpha(A) = \{x \in X / \mu_A(x) \geq \alpha\}.$$

Fuzzy množinu A nazveme konvexní, jestliže

$$\mu_A(\lambda \cdot x + (1-\lambda) \cdot y) \geq \min(\mu_A(x), \mu_A(y))$$

Fuzzy množinové operace

Při pojednání o fuzzy množinových operacích vyjdeme z operací množin klasických (ostrých). Budeme tedy hovořit o operaci průniku, sjednocení a doplňku.

V teorii klasických množin jsou množinové operace jednoduché a jednoznačné. Uvažujme dvě ostré množiny A a B definované na jediném univerzu X, popsané svými funkcemi příslušnosti podle Obr.5.1. Operace jejich průniku $A \cap B$, sjednocení $A \cup B$ a unární operaci doplňku A' můžeme vyjádřit vztahy

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)) \qquad (5.3)$$

$$\mu_{A'}(x) = 1 - \mu_A(x)$$

jejichž grafická interpretace je jasná.

Nyní rozšíříme svoje úvahy na operace nad dvěma fuzzy množinami A a B, definovanými opět na jednom univerzu X. Už ze samotného faktu, že funkce příslušnosti může nyní nabývat všech hodnot z intervalu ⟨0, 1⟩ vyplývá, že interpretace fuzzy množinových operací nebude tak jednoduchá a hlavně jednoznačná jako v případě operací nad množinami ostrými.

Předně je nutno uvést, že fuzzy množinová matematika připouští pro fuzzy množinové operace fuzzy průniku, fuzzy sjednocení a fuzzy doplňku tytéž vztahy, které jsou uvedeny v (5.3). Významné a pro fuzzy množinovou matematiku typické však je, že uvedené základní fuzzy množinové operace můžeme definovat i jinak, variantně. Další možné definice jsou např. tyto:

$$\mu_{A\cap B}(x) = \mu_A(x) \cdot \mu_B(x)$$

$$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$$

$$\mu_{A\cap B}(x) = \max(0, \mu_A(x) + \mu_B(x) - 1)$$

$$\mu_{A\cup B}(x) = \min(1, \mu_A(x) + \mu_B(x))$$

Výběr vhodné varianty operace pak záleží především na konkrétní řešené úloze. Naskýtá se otázka, jestli a když tak jakým způsobem lze třídy operací fuzzy průniku, fuzzy sjednocení či fuzzy doplňku definovat obecněji.

Třída operací, které splňují podmínky fuzzy průniku, se nazývá třída operací **triangulační normy** (t-normy), třída operací splňující podmínky fuzzy sjednocení se nazývá třídou operací **triangulační s-normy.**

Jsou-li dána čísla a,b,c,d,$\in$[0,1] potom t-norma

$$t(a,b) = a \hat{*} b$$

je binární operací, která představuje obecný operátor průniku a splňuje tyto axiomy

komutativnost	$a \hat{*} b = b \hat{*} a$
asociativnost	$(a \hat{*} b) \hat{*} c = a \hat{*} (b \hat{*} c)$
monotónnost	$a \le c \ \ and \ \ b \le d \Rightarrow a \hat{*} b \le c \hat{*} d$
hraniční podmínka	$a \hat{*} 1 = a$.

S-norma

$$s(a,b) = a \hat{*} b$$

je binární operací, která je zobecněným operátorem sjednocení a splňuje axiomy

komutativnost	$a \hat{*} b = b \hat{*} a$
asociativnost	$(a \hat{*} b) \hat{*} c = a \hat{*} (b \hat{*} c)$
monotónnost	$a \le c \ \ and \ \ b \le d \Rightarrow a \hat{*} b \le c \hat{*} d$
hraniční podmínka	$a \hat{*} 0 = a$

Operací, které mají charakter průniku fuzzy množin (t-norma) a sjednocení fuzzy množin (s-norma je celá řada [5].

Uveďme nyní další důležité operace nad fuzzy množinami. Je to především **omezený součet** dvou fuzzy množin A a B, jehož výsledkem je fuzzy množina definovaná vztahem

$$A \oplus B = \int \min(1, \mu_A(x) + \mu_B(x)) / x$$

kde + značí aritmetický součet. Další operací je **omezený rozdíl**, jehož výsledkem je fuzzy množina definovaná vztahem

$$A \Theta B = \int \max(0, \mu_A(x) - \mu_B(x)) / x$$

kde – značí aritmetický rozdíl. **Součinem fuzzy množin** je fuzzy množina definovaná vztahem

$$A \cdot B = \int_X \mu_A(x) \cdot \mu_B(x) / x$$

Fuzzy množinové operace – shrnutí

Pro pochopení využití fuzzy množin v metodách reprezentace znalostí a řídicích mechanizmů je důležité seznámení s fuzzy množinovými operacemi, které mají ve srovnání s operacemi klasických množin některá specifika. Kapitola podává vysvětlení definice fuzzy operací t-normy a s-normy, které mají mj. velký význam pro dosažení kvality přibližného usuzování ve fuzzy expertních systémech.

Kontrolní otázky:

1. Jaké jsou kvantitativní parametry pro vyjádření tvaru funkce příslušnosti fuzzy množiny?
2. Existuje paralela mezi operacemi s obyčejnými a fuzzy množinami?
3. Musí být konjunkce dvou fuzzy množin definovaných na jednom univerzu graficky interpretována vždy průnikem křivek jejich funkcí příslušnosti?

Otázka k zamyšlení:

1. Jaký je praktický důvod existence tříd funkcí t-norem a s-norem ve fuzzy množinové matematice?

Úkoly k zamyšlení:

Promyslete slovní interpretaci významu nekonvexní fuzzy množiny.

5.4 Fuzzy relace

Uvažujme dvě spočetná univerza U a V a fuzzy množinu danou funkcí příslušnosti

$\mu_R : U \times V \rightarrow [0, 1]$

která mapuje kartézský součin $U \times V$ na interval [0, 1]. Binární fuzzy relaci na kartézském součinu $U \times V$ definujeme vztahem

$$R = \sum_{U \times V} \mu_R(u,v)/(u,v)$$

V případě univerz nespočetných je tento vztah ve tvaru

$$R = \int_{U \times V} \mu_R(u,v)/(u,v)$$

n-ární fuzzy relaci definujeme jako fuzzy množinu n-tic s danými funkcemi příslušnosti

$$\mu_R(x_1, x_2, ...x_n)/(x_1, x_2, ...x_n)$$

Operace nad fuzzy relacemi

Uvažujme dvě binární relace R a S definované na kartézském součinu $X \times Y$. definujme nad těmito relacemi opět operace průniku a sjednocení.

Funkce příslušnosti výsledné fuzzy množiny průniku relací R a S S je definována vztahem

$$\mu_{R\cap S}(x,y) = \min(\mu_R(x,y), \mu_S(x,y))$$

pro všechna x, y. Uvědomíme si, že místo operace minima lze užít libovolné operace t-normy.

Funkce příslušnosti fuzzy množiny operace sjednocení relací R a S je dána vztahem

$$\mu_{R\cup S}(x,y) = \max(\mu_R(x,y), \mu_S(x,y))$$

pro všechna x, y. Místo operace maxima lze užít libovolné operace s-normy. Uvedené binární operace mohou být jednoduše rozšířeny na operace n-ární.

Významnou roli ve fuzzy množinové matematice zaujímají **operace projekce** a **operace cylindrické rozšíření**. Uvažujme binární relaci R definovanou na univerzu $X \times Y$. Pak definujeme projekci R na Y jako fuzzy množinu

$$proj\ R\ na\ Y = \int_Y \sup_{\forall x} \mu_R(x,y)/y$$

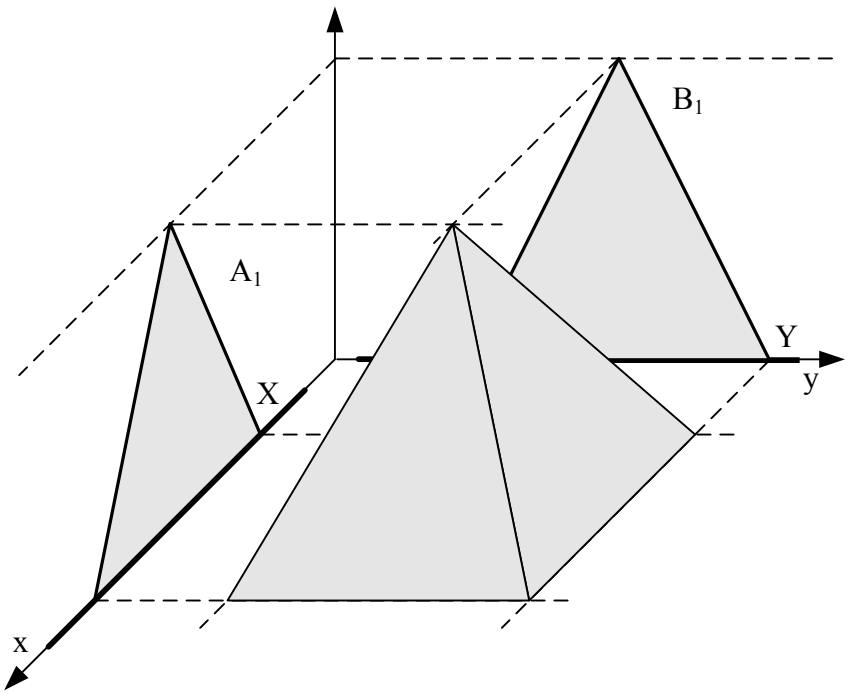
Operace projekce vytváří obecně z n-ární relace relaci l-ární, např. z ternární relace relaci binární, z binární relaci unární (obyčejnou fuzzy množinu).

Opačnou operací k operaci projekce je operace cylindrické rozšíření. Uvažujme fuzzy množinu F definovanou na univerzu Y. Cylindrickým rozšířením F na $X \times Y$ je množina všech dvojic $(x, y) \in X \times Y$ s funkcí příslušnosti danou vztahem

$$cyl(F) = \int_{X\times Y} \mu_F(y)/(x,y)$$

Uvažujme dvě fuzzy množiny A_1 a B_1, které jsou definovány na spojitých univerzech X, Y. Na Obr.5.13. je nakreslena konstrukce jejich fuzzy relace R, která vznikne jako průnik jejich cylindrických rozšíření.

Velmi důležitou fuzzy relací je **relace kompozice**. Kompozice je dána kombinací cylindricky rozšířené fuzzy množiny, fuzzy relace a následné projekce.



Obr.5.13

Uvažujme fuzzy množinu *A* definovanou na univerzu *X* a *R* je fuzzy relace definovaná na $X \times Y$. Potom kompozicí fuzzy množiny *A* a relace *R* je fuzzy množina *B*, definovaná na *Y*, pro kterou platí

$$B = A \circ R = proj\left(cyl\left(A\right) \cap R\right) na\, Y$$

Je-li průnik vytvořen pomocí operace *min* a projekce pomocí operace *max* pak funkce příslušnosti fuzzy množiny *B* jako kompozice *A* a *R* je dána vztahem

$$\mu_B(y) = \max_{\forall x} \min\left(\mu_A(x), \mu_R(x,y)\right)$$

a nazývá se **min-max** (minimaxová) kompozice. Je-li průnik vytvořen pomocí operace součinu a projekce pomocí operace *max* potom funkce příslušnosti relace kompozice má tvar

$$\mu_B(y) = \max_{\forall x}\left(\mu_A(x) \cdot \mu_R(x,y)\right)$$

a kompozice se nazývá **max-produkt** kompozicí.

Význam relace kompozice vyplyne v kapitole 5.7, věnované vyvozování závěrů expertních systémů.

Fuzzy relace - shrnutí

Kapitola je věnována operacím mezi fuzzy množinami, definovanými nad různými univerzy. Výsledné fuzzy relace mají velký význam při vysvětlení funkce procedur přibližného fuzzy usuzování. Jmenovitě jsou definovány důležité relace cylindrické rozšíření, projekce a kompozice.

Kontrolní otázky:

1. Jaká je grafická interpretace relace dvou fuzzy množin definovaných na různých univerzech?
2. Vysvětlete relaci kompozice fuzzy množin

Otázka k zamyšlení:

1. Jak lze znázornit operaci cylindrického rozšíření a projekce graficky?

5.5 Extenzionální princip

Extenzionální princip (princip rozšíření) je považován za jeden z nejefektnějších principů fuzzy množinové matematiky. Umožňuje totiž převést formálně jakoukoliv operaci mezi klasickými množinami na operaci mezi fuzzy množinami. Prakticky zvláště významná je aplikace tohoto principu na fuzzy množiny typu „fuzzy číslo“ (viz Obr.5.5), kdy můžeme hovořit o převedení operací nad obyčejnými čísly na operace nad fuzzy čísly. To umožňuje vytvořit fuzzy aritmetiku pro počítání s nepřesnými (fuzzy) čísly.

Uvažujme univerza U a V a funkci f, která mapuje U na V, tj

$f: U \rightarrow V$

a fuzzy množinu $A \subseteq U$. Fuzzy množina A pak ve V indukuje fuzzy množinu, jejíž funkce příslušnosti je dána vztahem

$$\mu_f(v) = \begin{cases} \sup_{f(u)=v} \mu_A(u) & jestliže\, existuje\ u \in U\ takové, že\ v = f(u) \\ 0 & jinak \end{cases}$$

Supremum se přitom bere přes všechna taková $u \subseteq U$, pro která je $f(u) = v$.
Princip rozšíření můžeme zobecnit i pro funkce více proměnných. Jsou-li např. U_1, U_2 univerza a A_1, A_2 na nich definované fuzzy množiny a je-li f binární funkce

$f: U_1 \times U_2 \rightarrow V$

potom

$$\mu_f(v) = \sup_{u_1,u_2 / f(u_1,u_2)=v} \min\left(\mu_{A_1}(u_1), \mu_{A_2}(u_2)\right). \tag{5.4}$$

Pomocí principu rozšíření lze definovat **aritmetiku fuzzy čísel**. Uveďme příklad součtu dvou fuzzy čísel *m* („asi *m*“) a *n* („asi *n*“). Tak např. podle (5.4) vypočítáme výsledné fuzzy číslo jejich **součtu** jako

$$\mu_{\tilde{m}\oplus\tilde{n}}(z) = \sup_{x,y / z=x+y} \min\left(\mu_{\tilde{m}}(x), \mu_{\tilde{n}}(y)\right)$$

Podobně jejich **rozdíl** je definován jako

$$\mu_{\tilde{m}\ominus\tilde{n}}(z) = \sup_{x,y / z=x-y} \min\left(\mu_{\tilde{m}}(x), \mu_{\tilde{n}}(y)\right)$$

a jejich **součin** vypočteme podle vztahu

$$\mu_{\tilde{m}\otimes\tilde{n}}(z) = \sup_{x,y / z=x\cdot y} \min\left(\mu_{\tilde{m}}(x), \mu_{\tilde{n}}(y)\right)$$

Jak vidíme, princip rozšíření je velmi užitečný nástroj, který umožňuje jednoduchou fuzzifikaci klasických matematických vztahů.

Extenzionální princip - shrnutí

Extenzionální princip umožňuje rozšíření aritmetiky obyčejných čísel na aritmetiku fuzzy čísel. Umožňuje definici fuzzy algebry a realizaci početních operací s fuzzy čísly.

Kontrolní otázky:

1. Proč je ve vztahu formalizující extenzionální princip použit symbol suprema? Proč nestačí maximum?

Otázka k zamyšlení:

1. Jaký je hlavní význam extenzionálního principu ve fuzzy množinové matematice?

Úkoly k zamyšlení:

Může být výsledkem binární fuzzy operace mezi dvěma konvexními normálními fuzzy množinami fuzzy množina nekonvexní, která navíc nemá vlastnost normality?

5.6 Základy fuzzy logiky

Fuzzy pravidla a fuzzy výroky

Jádrem expertní systémů je báze znalostí, v níž jsou uloženy formalizované expertní znalosti problémové oblasti, v níž má expertní systém poskytovat podporu při rozhodování. Fuzzy expertní systémy používají pro formalizaci znalostí známých IF-THEN pravidel ve formě

IF(*předpoklad, antecedent*) THEN (*důsledek, konsekvent*)
(5.5)

Antecedenty i konsekventy pravidel jsou vágní **fuzzy výroky** - jejich pravdivostní hodnota leží v intervalu $\langle 0, 1 \rangle$.

Struktura fuzzy výroků obsahuje jazykové proměnné, jejich jazykové hodnoty a fuzzy logické spojky. Typickou formou je výraz

(X is A)

kde X je **jazyková proměnná** a A je její příslušná **jazyková hodnota**. Vágnost jazykových hodnot je přitom formalizována fuzzy množinami.

Jazykovou proměnnou lze popsat uspořádanou čtveřicí

$$\langle X, \text{Ł}X, U, M_X \rangle$$

kde X je jméno jazykové proměnné, $\text{Ł}X = \{LX_1, LX_2, \ldots, LX_n\}$ je množina jejich jazykových hodnot, U je univerzum a M_X je funkce, vyjadřující význam jazykových hodnot pomocí fuzzy množin.

Složené fuzzy výroky

Uvažujme dva fuzzy výroky p: (X is A) a q: (Y is B) , kde A, B jsou fuzzy množiny definované na jednom univerzu U. Takové výroky nazýváme **atomické**. Atomické výroky mohou být spolu spojovány fuzzy logickými spojkami *and, or, not* a mohou tak tvořit fuzzy výroky **složené**. Těmto spojkám odpovídá konjunkce, disjunkce resp. negace fuzzy množin, jak bylo uvedeno výše.

Jestliže p a q jsou atomické fuzzy výroky p: (X is A) a q: (Y is B) , kde A, B jsou fuzzy množiny definované na jednom univerzu U, potom význam složeného výroku

(X is A) *and* (Y is B)

je dán konjunkcí fuzzy množin $A \cap B$, kde fuzzy konjunkce může být interpretována jakoukoliv t-normou.

Jestliže p a q jsou atomické fuzzy výroky fuzzy výroky p: (X is A) a q: (Y is B) , kde A, B jsou fuzzy množiny definované na jednom univerzu U, potom význam složeného výroku

(X is A) *or* (Y is B)

je dán disjunkcí fuzzy množin $A \cup B$, kde fuzzy disunkce může být interpretována jakoukoliv s-normou.

Negace atomického výroku (X is A) je výrok (X is not A) a je dána doplňkem A' fuzzy množiny A podle vztahu (5.3).

Předpokládejme nyní, že dvě jazykové proměnné E, E' jsou definovány na dvou různých univerzech U, U'. Dále předpokládejme, že platí dva fuzzy výroky p a q, kde p: (E is A) a q: (E' is B). Jazykové hodnoty A a B jsou reprezentovány fuzzy množinami A a B:

$$A = \int_U \mu_A / e \qquad B = \int_{U'} \mu_B(e') / e'$$

Význam složeného výroku

(E is A) *and* (E' is B) (5.6)

je třeba interpretovat tak, že nejprve musíme provést cylindrické rozšíření fuzzy množin A a B na kartézský součin $U \times U'$. Pak význam složeného výroku (5.6) je reprezentován fuzzy relací, definovanou na tomto kartézském součinu podle vztahu

$$\mu_r(e,e') = \int_{U\times U'} \min(\mu_A(e), \mu_B(e'))/(e,e')$$

kde operátor *min* může být nahrazen libovolnou t-normou.

Význam složeného výroku

(E is A) *or* (E' is B)

je opět reprezentován fuzzy relací na univerzu $U \times U'$, vyjádřenou nyní vztahem

$$\mu_s(e,e') = \int_{U\times U'} \max(\mu_A(e), \mu_B(e'))/(e,e')$$

a operátor *max* může být opět nahrazen libovolnou s-normou.

Všimneme si nyní důležitého fuzzy logického funktoru THEN, spojujícího v pravidle

IF (*fuzzy výrok – atomický nebo složený*) THEN (*fuzzy výrok – atomický nebo složený*)

antecedent a konsekvent. V souladu s významem pravidla (5.5) interpretujeme spojku THEN obecně jako implikaci. Pravidlo IF-THEN tak vyjadřuje kauzální vztah mezi výroky v antecedentu a konsekventu.

Tento kauzální vztah představuje relaci mezi jazykovými proměnnými v antecedentu a konsekventu. Takovou relaci je možno v našem případě vyjádřit ve tvaru fuzzy relace. V případě fuzzy IF-THEN pravidla je tato relace reprezentována **fuzzy implikací**.

Narozdíl od klasické logiky není interpretace fuzzy implikace (podobně jako tomu bylo v případě fuzzy konjunkce, fuzzy disjunkce a fuzzy negace) jednoznačná. Při její konstrukci vyjdeme z analogie ve dvouhodnotové logice.

Uvažujme dva klasické (ostré) výroky *p* a *q*. Jejich implikaci

$p \rightarrow q$

je v klasické logice možno vyjádřit pomocí jednoduchých logických funkcí vztahem

$$p \rightarrow q = (not\ p) or\ q \tag{5.7}$$

nebo

$$p \rightarrow q = (p\ and\ q)\ or\ (not\ p) \tag{5.8}$$

Uvažujeme-li nyní výroky *p* a *q* jako fuzzy výroky, můžeme pro operaci *and* a *or* použít libovolné t-normy resp. s-normy. Podle konkrétně zvolené normy pak získáme celou řadu možných interpretací fuzzy implikace (někdy nazývané fuzzy implikačními funkcemi [60]).

Předpokládejme, že fuzzy výrok *p* obsahuje fuzzy množinu $A \subseteq U_A$ a fuzzy výrok *q* fuzzy množinu $B \subseteq U_B$. Před vlastní aplikací příslušných norem je opět nutno provést cylindrické rozšíření fuzzy množin *A* a *B* na dvojrozměrné univerzum $U_A \times U_B$. V současné době je tak známo asi 40 druhů různých fuzzy implikací [5]. Uveďme alespoň nejvýznamnější z nich.

Implikace Kleene-Dienesova (booleovská)

V interpretaci této implikace je ve vztahu (5.7) použito pro funktor *not* operátoru (1 – a) a pro funktor *or* s-normy *max.* Před aplikací operátoru *or* je opět třeba provést cylindrické rozšíření.

$$R_b = cyl(A') \cup cyl(B) = \int_{X \times Y} \max(1 - \mu_A(x), \mu_B(y))/(x, y)$$

$$\mu_{R_b}(x, y) = \max(1 - \mu_A(x), \mu_B(y))$$

Implikace Lukasiewiczova

V interpretaci této implikace je ve vztahu (5.7) použito pro funktor *not* operátoru (1 – a) a pro funktor *or* s-normy *omezený součet.* Před aplikací operátoru *or* je opět třeba provést cylindrické rozšíření.

$$R_a = cyl(A') \oplus ce(B) = \int_{X \times Y} \min(1, 1 - \mu_A(x) + \mu_B(y))/(x, y)$$

$$\mu_{R_a}(x, y) = \min(1, 1 - \mu_A(x) + \mu_B(y))$$

Implikace Zadehova

V interpretaci této implikace je ve vztahu (5.8) použito pro funktor *not* operátoru (1 – a), pro funktor *and* je použito t-normy *min* a pro operaci *or* je využito s-normy *max.*

$$R_m = (cyl(A) \cap cyl(B)) \cup cyl(A') = \int_{X \times Y} \max(\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x))/(x, y)$$

$$\mu_{R_m}(x, y) = \max(\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x))$$

Implikace Larsenova

Využívá vztahu $p \rightarrow q$ = (*p and q*), kde pro operaci *and* je použito t-normy *algebraický součin*. Z matematického hlediska se pro tuto relaci nejedná o implikaci, je však v literatuře takto nazývána a pro její funkci příslušnosti platí

$$R_p = cyl(A) \cap cyl(B) = \int_{X \times Y} \mu_A(x) \cdot \mu_B(x)/(x, y)$$

$$\mu_{R_p}(x, y) = \mu_A(x) \cdot \mu_B(y)$$

Implikace Mamdaniho

Tato velmi často používaná implikace využívá opět vztahu $p \rightarrow q$ = (*p and q*), v němž operace *and* je reprezentována t-normou *min.* Z matematického hlediska se opět nejedná o implikaci.

$$R_c = cyl(A) \cap cyl(B) = \int_{X \times Y} \min(\mu_A(x), \mu_B(y))/(x, y)$$

$$\mu_{R_c}(x, y) = \min(\mu_A(x), \mu_B(y))$$

Základy fuzzy logiky - shrnutí

Fuzzy logika, využívající fuzzy množin, je teoretickým zázemím systémů pro přibližné fuzzy usuzování. Zavádí pojem fuzzy výroků a definuje postupy jejich využití ve fuzzy podmíněných *IF-THEN* pravidlech. V kapitole jsou uvedeny přístupy k interpretaci jednoduchých logických funkcí fuzzy disjunkce, konjunkce a negace. Zvláštní pozornost je pak věnována interpretaci složené funkce fuzzy implikace, jsou uvedeny její v praxi nejpoužívanější varianty.

Kontrolní otázky:

1. Jaký je rozdíl mezi výrokem v klasické a fuzzy logice?
2. Co je to složený fuzzy výrok a jaký má význam?
3. Proč existuje celá třída fuzzy implikací?

Otázka k zamyšlení:

1. Proč existuje celá třída fuzzy implikací?

Úkoly k zamyšlení:

Grafickým způsobem znázorněte, jaký je rozdíl mezi Mamdaniho a Larsenovou implikací.

5.7 Přibližné usuzování

Jak již bylo uvedeno výše, fuzzy implikace

IF(x is A_1) THEN (y is B_1)

vyjadřuje kauzální vztah mezi jazykovými proměnnými x a y. Jinými slovy – jestliže jazyková proměnná x nabude své jazykové hodnoty A_1, důsledkem je stav, kdy jiná jazyková proměnná y nabude své jazykové hodnoty B_1. Toto pravidlo můžeme podle výše uvedených zásad vyjádřit jako fuzzy relaci R.

Uvažujme nyní případ, kdy víme že jazyková proměnná x nabude své jiné jazykové hodnoty, např. A_2, tedy (x is A_2). Ptáme se, jaké jazykové hodnoty nabude nyní jazyková proměnná y ?

K odpovědi na tuto otázku použijeme procesu tzv. **přibližného usuzování** (aproximativní inference). Jde zřejmě o postup logického usuzování a proto k jeho konstrukci použijeme již dříve zmíněného klasického pravidla usuzování modus ponens. Toto pravidlo nám umožňuje usuzovat na pravdivost výroku **B** na základě pravdivosti výroku **A**. Jeho zápis má obvykle tuto formu [2]:

Podmínka: Je-li pravdivé **A** pak je také pravdivé **B**
Premisa: **A** je pravdivé
Závěr: **B** je pravdivé

Pravdivost výroků **A** a **B** může přitom nabývat hodnot 0 (zcela nepravdivý) nebo 1 (zcela pravdivý).

Uvažme nyní, že výroky **A** a **B** jsou fuzzy výroky A, B a dále existují dva další modifikované fuzy výroky A_m a B_m. Dále nechť platí tzv**. zobecnělé (fuzzy) pravidlo modus ponens**, jehož zápis bude využívat forem fuzzy výroků a bude mít nyní tvar

Podmínka: Jestliže (x je A) pak (y je B)
Premisa: x je A_m (5.12)
Závěr: y je B_m

Jako ilustraci uveďme často uváděný příklad [2]

Podmínka: Jestliže je rajské jablíčko červené pak je zralé
Premisa: Jablíčko je velmi červené
Závěr: Jablíčko je velmi zralé

Fuzzy výroky A a B byly v našem příkladu modifikovány jazykovým operátorem „velmi“. Uveďme [2], že závěr přibližného úsudku (tj. tvar fuzzy množiny B_m) lze získat pomocí operace kompozice

$$B_2 = A_2 \circ R = proj\left(cyl\left(A_2\right) \cap R\right) \ na \ Y$$

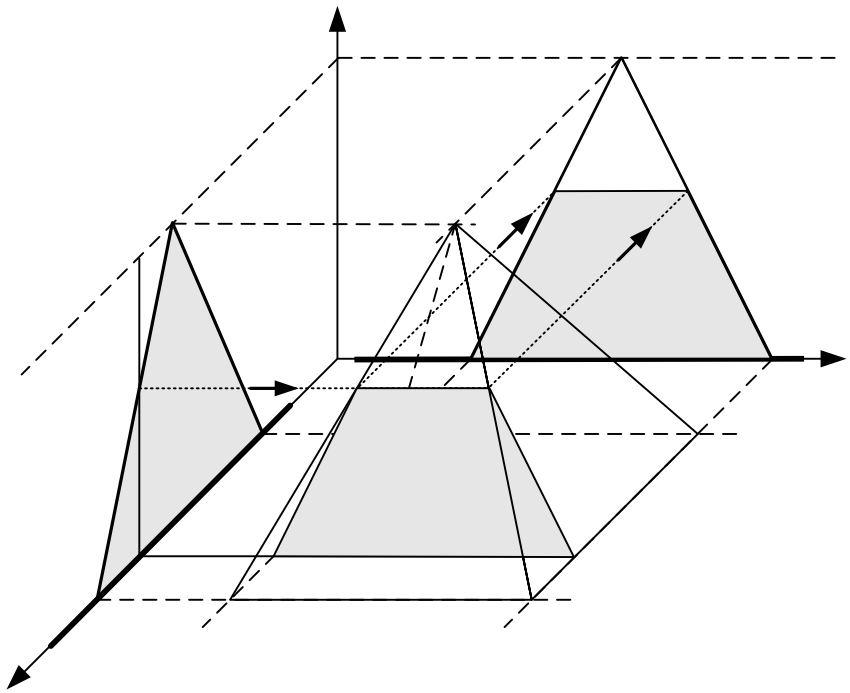
kde R je fuzzy relace, reprezentující fuzzy implikaci

IF(x is A_1) THEN (y is B_1)

V případě, že tato implikace je typu Mamdani, platí pro funkci příslušnosti výsledné fuzzy množiny

$$\mu_{B_2}(y) = \sup_x \left\{ \min\left[\mu_{A_2}(x), \mu_R(x,y) \right] \right\} \tag{5.13}$$

Velmi častým případem je situace, kdy modifikovaná fuzzy množina A_m (5.12) je fuzzy množina typu singleton – ostré číslo a fuzzy množiny A_1, B_1 jsou definovány na spojitých univerzech. V případě použití Mamdaniho implikace je možno proceduru přibližného usuzování (vyvození závěru B_m znázornit graficky, jak je uvedeno na Obr.5.14 (srovnej konstrukci fuzzy relace R s konstrukcí na Obr.5.13!).

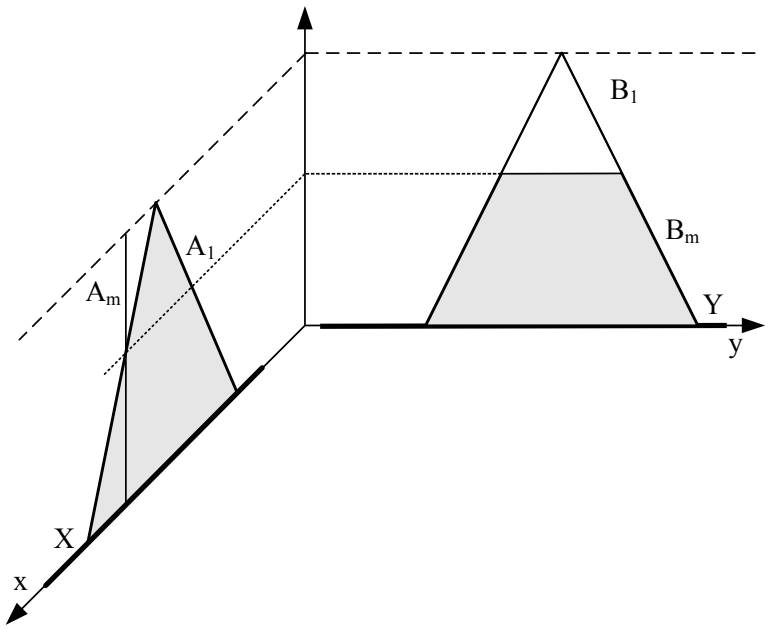


Obr.5.14

Supremum v rovnici (5.13) tvoří ostrá hodnota x^* a pro funkci příslušnosti výstupní fuzzy množiny B_2 platí

$$\mu_{B_2} = \left\{\min\left[\mu_{A_1}(x^*), \mu_R(x^*, y)\right]\right\} = \mu_R(x^*, y)$$

Rovina, proložená vstupním singletonem x* vytne v relaci *R* lichoběžník, který je pak promítnut operací projekce na univerzum *Y*. V jedné rovině pak lze Obr.5.14 nakreslit jednodušeji - viz Obr.5.15.



Obr.5.15

Přibližné usuzování - shrnutí

Kapitola uvádí problematiku zobecnění klasického pravidla usuzování modus ponens do jeho fuzifikované firmy. Představuje přístup použití takového zobecnělého pravidla pro metodiku stanovení fuzzy množiny jako závěru přibližného usuzování s využitím kompozičního pravidla. Metoda je základem řídicích mechanizmů fuzzy orientovaných expertních systémů.

Kontrolní otázky:

1. Jaká je formulace pravidla usuzování fuzzy modus ponens?
2. Jaká je grafická interpretace procesu usuzování pomocí pravidla fuzzy modus ponens?

Otázka k zamyšlení:

1. Jaký je rozdíl mezi obyčejným a zobecnělým usuzovacím pravidlem modus ponens?

Úkoly k zamyšlení:

Promyslete vlastní příklad usuzování pomocí pravidla fuzzy modus ponens.

5.8 Fuzzy model jako báze znalostí

Dosud bylo středem našeho zájmu jediné pravidlo, představující fuzzy implikaci, reprezentované fuzzy relací *R*. Ukázali jsme, jak lze vyvodit tvar fuzzy množiny jazykové proměnné konsekventu při modifikované velikosti jazykové proměnné antecedentu.

Jediné pravidlo s atomickým fuzzy výrokem o velikosti jediné jazykové proměnné v antecedentu a jedním atomickým fuzzy výrokem o velikosti jiné jazykové proměnné v konsekventu zřejmě nestačí pro popis chování složité soustavy, která je obvykle pomocí expertního systému vyšetřována. Proto musíme k popisu chování takové soustavy použít více pravidel, které tvoří vlastní bázi znalostí a jsou v ní obsaženy všechny znalosti (experta), které máme o chování soustavy k dispozici.

Uvažujme tedy obecně jazykové proměnné $\langle X, ŁX, U_X, M_X \rangle$ a $\langle Y, ŁY, U_Y, M_Y \rangle$ a soubor *n* pravidel, které lze napsat ve tvaru

$$R_m^{(k)} : if \left(X \; je \, LX^{(k)}\right) \; then \quad \left(Y \; je \, LY^{(k)}\right) \quad k = 1,2,...n$$

kde $ŁX^{(k)} \in ŁX$ je jazyková hodnota proměnné X v antecedentu *k*-tého pravidla a $ŁY^{(k)} \in ŁY$ je jazyková hodnota proměnné Y v konsekventu *k*-tého pravidla. Uvažujme dále, že každé pravidlo představuje implikaci typu Mamdani reprezentovanou fuzzy relací

$$R_m^{(k)} = \int_{U_X \times U_Y} \min\left(\mu_{LX^{(k)}}(x), \mu_{LY^{(k)}}(y)\right)/(x,y) \quad k = 1,2,...n$$

s funkcí příslušnosti

$$\mu_{R_m^{(k)}}(x,y) = \min\left(\mu_{LX^{(k)}}(x), \mu_{LY^{(k)}}(y)\right)$$

Předpokládejme, že jednotlivá pravidla jsou v souboru pravidel (bázi znalostí) spojena fuzzy logickým operátorem *or* s interpretací *max*. Takový soubor nazýváme **součtovým fuzzy modelem** a výslednou relaci, která zohledňuje dílčí výsledné relace jednotlivých pravidel dostaneme sjednocením

$$R_m = \bigcup_{k=1}^{n} R_m^{(k)}$$

s výslednou funkcí příslušnosti

$$\mu_{R_m}(x,y) = \max_{\forall k} \mu_{R_m^{(k)}}(x,y) = \max_{\forall k} \min\left(\mu_{LX^{(k)}}(x), \mu_{LY^{(k)}}(y)\right)$$

Poznamenejme, že pokud je funktor spokující pravidla typu *and* nazýváme takový fuzzy model **modelem součinovým** [6], [7].

V procesu aproximativního vyvozování pak budeme uvažovat, že jazyková proměnná X nabývá modifikované hodnoty reprezentované fuzzy množinou X_1. Tuto fuzzy množinu můžeme považovat z hlediska fuzzy modelování za **fuzzy množinu vstupní**. Ta indukuje prostřednictvím fuzzy relace R_m fuzzy množinu Y_1, která je výsledkem procedury přibližného úsudku a je možno ji považovat za **fuzzy množinu výstupní**. Tu získáme – jak jsme ukázali výše - pomocí operace kompozice fuzzy množiny X_1 a výsledné relace R_m , tedy

$$Y_1 = X_1 \circ R_m = proj\left(ce(X_1) \cap R_m\right) na\ Y$$

s funkcí příslušnosti

$$\mu_{Y_1}(y) = \max_{\forall x}\left(\min\left(\mu_{X_1}(x), \mu_{R_m}(x,y)\right)\right)$$

S uvážením existence spojovacího funktoru *or* mezi pravidly můžeme popsat proceduru aproximativního vyvození vztahy

$$Y_1^{(k)} = X_1 \circ R_m^{(k)} = proj\left(ce(X_1) \cap R_m^{(k)}\right) na\ Y \quad k = 1,2,...n$$

$$\mu_{Y_1^{(k)}}(y) = \max_{\forall x}\left(\min\left(\mu_{X_1}(x), \mu_{R_m^{(k)}}(x,y)\right)\right) \quad k = 1,2,...n$$

$$Y_1 = \bigcup_{k=1}^{n} Y_1^{(k)} = \bigcup_{k=1}^{n} X_1 \circ R_m^{(k)}.$$

V obecnějším případě budeme uvažovat soubor pravidel s implikací typu Mandami, vzájemně spojených fuzzy logickou spojkou *or*. Antecedent pravidel však nebude tvořen jediným atomickým fuzzy výrokem o velikosti jediné jazykové (vstupní) proměnné X, nýbrž bude tvořen výrokem složeným, popisujícím velikost n- jazykových proměnných vázaných mezi sebou fuzzy logickým operátorem *and* (.):

$$IF\left(x_1 isA_1^{j_r}\right) and...and\left(x_n isA_n^{j_r}\right) THEN\ \left(y isB^{j_r}\right) \quad r = 1,2,...R$$

kde r je číslo pravidla a R je celkový počat pravidel souboru. Operátor *and* je dán konjunkcí fuzzy výroků, kde fuzzy konjunkce může být interpretována jakoukoliv t-normou. V modelu Mandami je interpretována jako *min.*

Fuzzy model jako báze znalostí - shrnutí

Kapitola představuje množinu fuzzy podmíněných pravidel jako součtový nebo součinový jazykový fuzzy model. Ten je prostředkem formalizace neurčitých znalostí v bázi znalostí expertního systému. Zvláštní pozornost je věnována fuzzy modelu typu Mamdani. Je uveden ilustrativní příklad fuzzy modelování v systému se dvěma vstupními a jednou výstupní proměnnou.

Kontrolní otázky:

1. Jaká je souvislost mezi fuzzy modelem a bází znalostí expertního systému?
2. Čím se vyznačuje fuzzy model typu Mamdani? Proč se tak nazývá?

Otázka k zamyšlení:

1. Má praktické využití fuzzy model, jehož pravidla jsou spojena logickou spojkou fuzzy konjunkce?

Úkoly k zamyšlení:

Promyslete jednoduchou bázi znalostí pro řešení problému využití volného času.

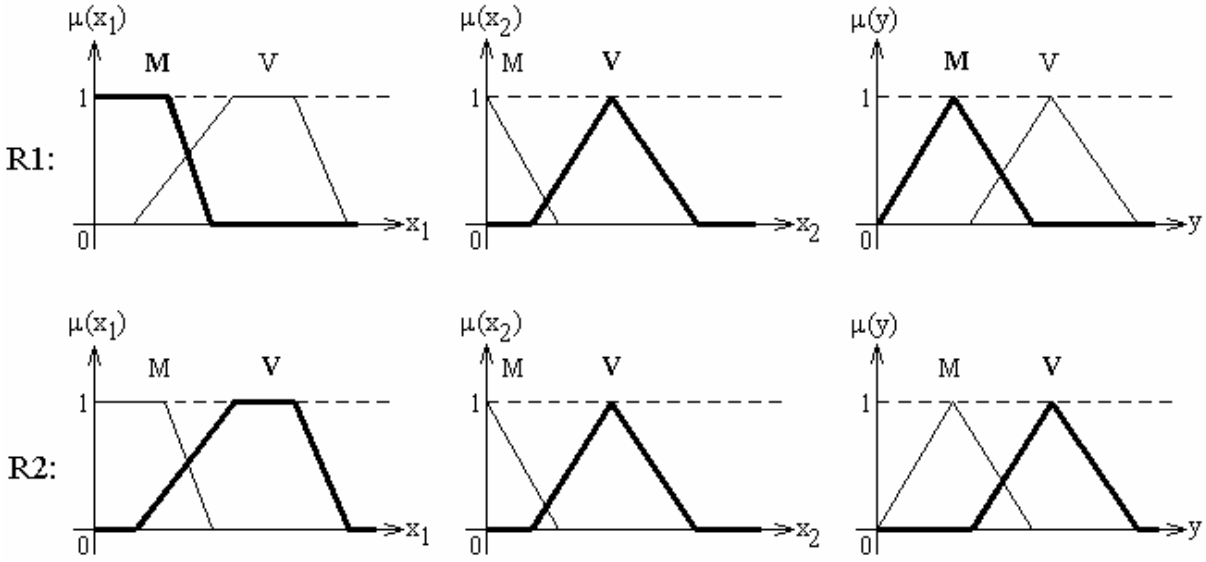
5.8.1 Fuzzy model typu Mamdani

Uvažujme soustavu s dvěma vstupními a jednou výstupní proměnnou [6]. Tuto soustavu budeme modelovat s využitím fuzzy modelu Mamdani, který bude mít dvě vstupní a jednu výstupní jazykovou proměnnou. Všechny proměnné nechť mají dvě jazykové hodnoty – *MALÝ* a *VELKÝ*.

Sestavíme jazykový model soustavy s použitím dvou pravidel, z nichž první se bude vyjadřovat k situaci, v níž je hodnota výstupní proměnné *MALÁ* a druhé k situaci v níž je hodnota výstupní proměnné *VELKÁ*.

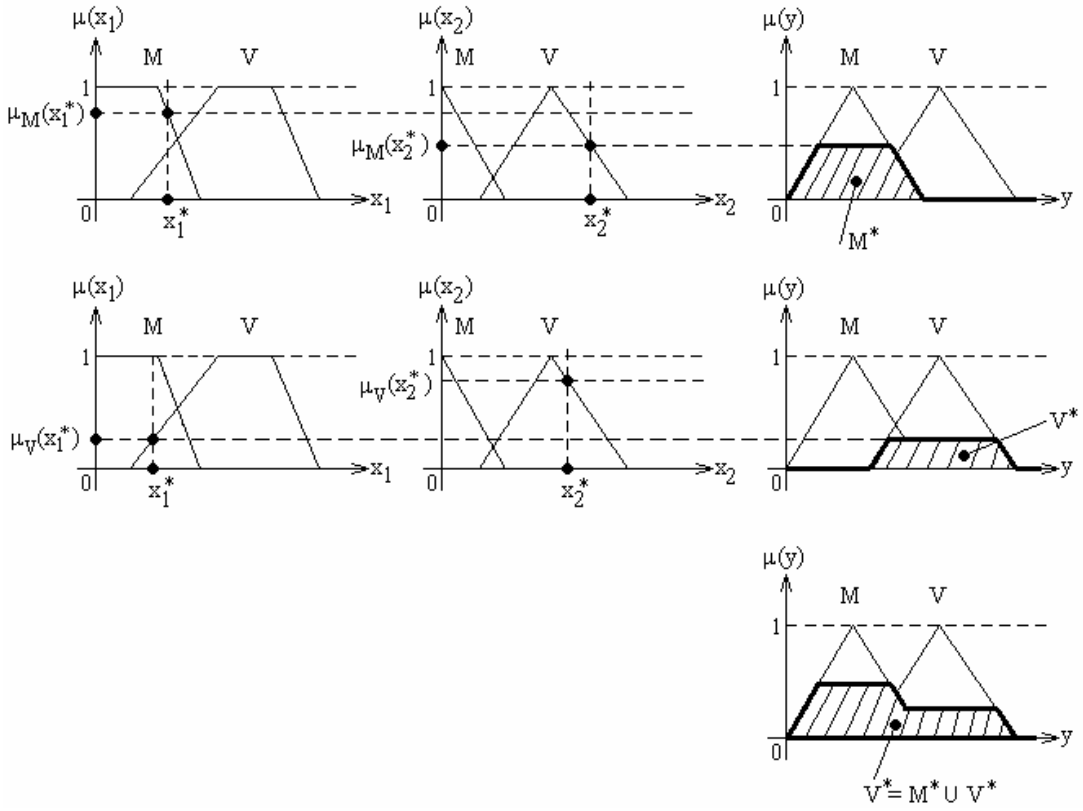
R_1: IF(x_1 is *MALÝ*) and (x_2 is *VELKÝ*) THEN (y is *MALÝ*)
R_2: IF(x_1 is *VELKÝ*) and (x_2 is *VELKÝ*) THEN (y is *VELKÝ*)

Jazykové proměnné vstupních veličin x_1 a x_2 a výstupní veličiny y jsou nakresleny na obrázku Obr.5.16, který současně reprezentuje graficky obě pravidla modelu.



Obr.5.16

Fuzzy množiny jazykových hodnot, které jsou aktuální v jednotlivých pravidlech jsou zdůrazněny tučnými čarami.



Obr.5.17

Stanovení tvaru výsledné fuzzy množiny výstupní proměnné *y* při uvažování (dosazení) konkrétních hodnot vstupních proměnných x_1 a x_2 postupujeme podle obrázku Obr.5.17 takto:

- dosazované konkrétní hodnoty vstupních proměnných jsou označeny x_1^* a x_2^*.

- nejprve zjistíme tvar výstupní fuzzy množiny pro první pravidlo. V souladu se zněním prvního pravidla zjistíme velikost stupně příslušnosti hodnoty x_1^* k fuzzy množině *M* a označíme jej jako $\mu_M(x_1^*)$. Dále zjistíme velikost stupně příslušnosti hodnoty x_2^* k fuzzy množině V a označíme jej jako $\mu_V(x_2^*)$.

- porovnáme velikosti $\mu_M(x_1^*)$ a $\mu_V(x_2^*)$ a vybereme menší z nich, tedy $\mu_V(x_2^*)$. Touto hodnotou „ořežeme“ jazykovou hodnotu výstupní hodnoty *y* v prvním pravidle, tedy fuzzy množinu *M*. Ořezanou fuzzy množinu označíme jako M^* a budeme ji uvažovat jako **výstup prvního pravidla**.

- nyní zjistíme tvar výstupní fuzzy množiny pro druhé pravidlo. V souladu se zněním druhého pravidla zjistíme velikost stupně příslušnosti hodnoty x_1^* k fuzzy množině *V* a označíme jej jako $\mu_V(x_1^*)$. Dále zjistíme velikost stupně příslušnosti hodnoty x_2^* k fuzzy množině *V* a označíme jej jako $\mu_V(x_2^*)$.

- porovnáme velikosti $\mu_V(x_1^*)$ a $\mu_V(x_2^*)$ a vybereme menší z nich, tedy $\mu_V(x_1^*)$. Touto hodnotou „ořežeme“ jazykovou hodnotu výstupní hodnoty *y* v druhém pravidle, tedy fuzzy množinu *V*. Ořezanou fuzzy množinu označíme jako V^* a budeme ji uvažovat jako **výstup druhého pravidla**.

- **výslednou výstupní fuzzy množinu** s uvažováním obou pravidel Y^* získáme sjednocením fuzzy množin M^* a V^*

$$Y^* = M^* \cup V^*$$

Výsledkem přibližného usuzování je tedy fuzzy množina *Y*,* kterou můžeme jazykově interpretovat jako „*SPÍŠE MALÝ*“.

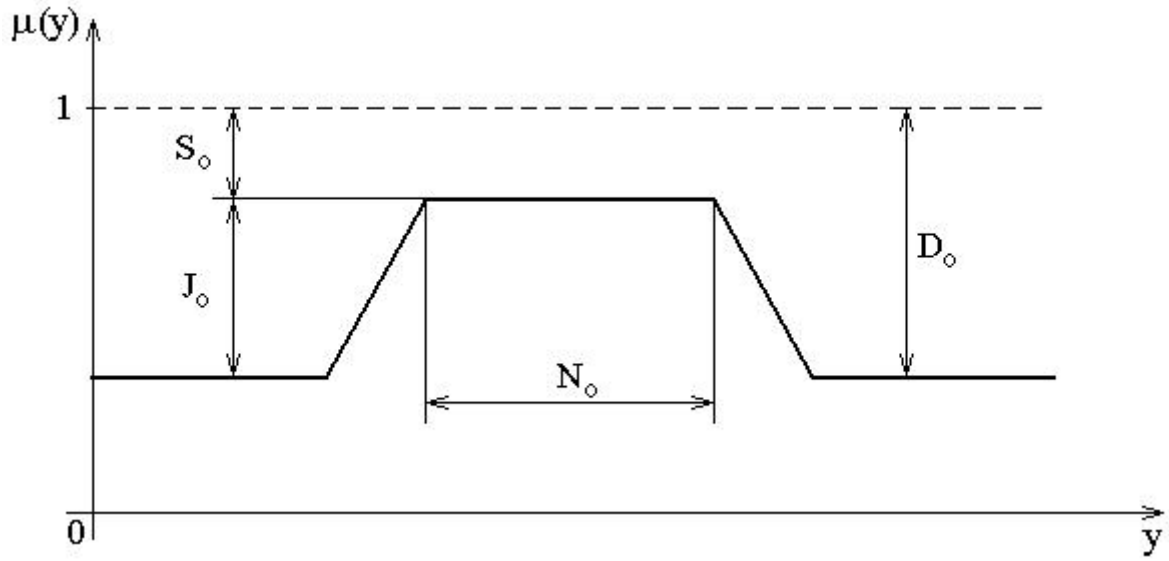
Popsaná metoda vyvozování se nazývá Zadehova interpolační metoda [6]. Pokud bychom potřebovali výstup ve formě čísla, provedeme tzv. **defuzzifikaci fuzzy množiny *Y**** podle metod, popsaných v kap.5.10.

5.9 Interpretace výsledků přibližného úsudku

Výstupem expertního systému – použijeme-li terminologie konzultace uživatele s expertem, tedy „odpověď“experta uživateli na jeho položený „dotaz“ – má tvar fuzzy množiny. Obecně nemusí být konvexní ani normální. Z hlediska uživatele je pak velmi důležitá správná interpretace významu takové fuzzy množiny jako vyvozené odpovědi [6].

V dalším budeme uvažovat, že báze expertního systému je složena z pravidel typu Mamdani spojených fuzzy logickým operátorem *or* (interpretovaným jako *max*), fuzzy logické operátory mezi dílčími fuzzy výroky antecedentů pravidel *and* (interpretovány jako *min*) a fuzzy implikace *THEN* mezi antecedenty a konsekventy pravidel je interpretována jako *min* (takový model se označuje jako model *CCD* – Conjunction-Conjunction-Disjunction [7]).

K posouzení kvality výstupní fuzzy množiny jako odpovědi modelu vzhledem k očekávání uživatele by bylo velmi výhodné, kdyby bylo možno stanovit její kvalitativní parametry. Kvalitu odpovědi lze parametrizovat obtížně, pokus uvádí [6]. Parametry kvality součtového modelu *CCD* jsou definovány podle Obr.5.18 takto:

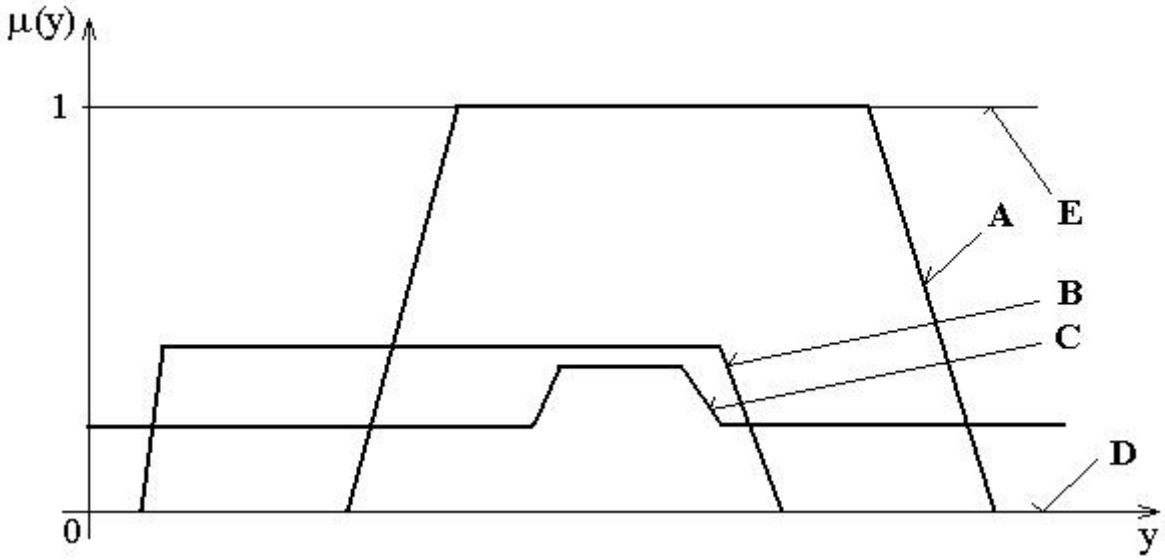


Obr.5.18

1. Spornost odpovědi	$S_o = 1 - \max\mu(y)$
2. Důvěryhodnost použitých informací	$D_o = 1 - \min\mu(y)$
3. Neurčitost odpovědi	$N_o = b - a$
4. Jasnost odpovědi	$J_o = \max\mu(y) - \min\mu(y)$

Interpretace typických odpovědí pro model *CCD* jsou uvedeny na Obr.5.19. Slovně je lze interpretovat takto:

A - zcela jasná odpověď,
B - odpověď při existenci sporu mezi pravidly,
C - odpověď při nedostatku informací.



Obr.5.19

Je-li nutno převést fuzzy množinu odpovědi do formy obyčejného čísla, použijeme proceduru defuzzifikace, jak bude vysvětleno v kap.5.10.

Interpretace výsledků přibližného úsudku - shrnutí

V případech konzultačních expertních systémů má velkou důležitost správná slovní interpretace tvarů fuzzy množin jejich závěrů uživatelem. Kapitola uvádí přístup kvantitativní parametrizace tvaru fuzzy množiny ve spojitosti s jejím významem jako odpovědi systému. Podává interpretaci nejčastěji se vyskytujících tvarů fuzzy množin jako odpovědí expertních systémů s bázemi znalostí ve formě Mamdaniho modelu. Je uveden příklad jednoduchého fuzzy expertního systému vytvořeného v prostředí LMPS.

Kontrolní otázky:

1. Jaké jsou parametry hodnocení kvality fuzzy množiny jako odpovědi expertního systému?
2. Jak jsou interpretovány odpovědi fuzzy systému, jehož výstupní jazyková proměnná má jazykové hodnoty reprezentované singletony?

Otázka k zamyšlení:

1. Jak je interpretována odpověď ve formě fuzzy množiny jejíž funkce příslušnosti má nekonečně velké jádro?

Úkoly k zamyšlení:

Jak je třeba upravit bázi znalostí, dává-li expertní systém odpovědi ve formě nekonvexní fuzzy množiny?

5.9.1 Fuzzy orientovaný expertní systém

Pro ilustraci fuzzy expertního systému je uveden příklad odhadu zbytkové aktivity katalyzátoru při výrobě kyseliny sírové.

V čem spočívá složitý problém, spojený s tímto technologickým procesem? Výroba kyseliny sírové probíhá prostřednictvím chemických reakcí uvnitř uzavřeného chemického reaktoru. Chemické reakce jsou usměrňovány katalyzátorem, který se do reaktoru spolu se vstupními surovinami přidá. V průběhu technologického procesu probíhajícího v reaktoru se katalyzátor vyčerpává, jeho chemická aktivita klesá a čas od času je vyčerpaný katalyzátor potřeba nahradit katalyzátorem čerstvým. Potíž spočívá v tom, že míru vyčerpání katalyzátoru (tedy jeho okamžitou zbytkovou aktivitu) nelze měřit. Zkušený provozní operátor umí míru jeho vyčerpanosti pouze odhadovat podle různých hledisek. To je typická situace - myšlenkový postup takového operátora lze dobře nahradit expertním systémem.

Zkušenosti ukazují, že míra vyčerpání katalyzátoru (velikost jeho zbytkové aktivity) je určena především těmito hledisky:

doba jeho použitím v reaktoru
provozní teplota v reaktoru
počet zastávek ve výrobě
průměrná délka zastávky.

Tato hlediska jsou i neodborníkovi zřejmá. Formulujme nyní dosavadní poznatky do zadání pro vytvoření odpovídajícího expertního systému. Vstupními proměnnými systému budou vyjmenovaná hlediska, výstupní proměnnou pak velikost odhadované zbytkové aktivity. Jde o model se čtyřmi vstupními (x_1, x_2, x_3, x_4) a jednou výstupní (y) jazykovou proměnnou. Spolu s rozměry je budeme definovat tabulkou na Obr.5.20.

Č.	**Jazyková proměnná**	**Rozměr**
x_1	Doba provozu katalyzátoru	měsíc
x_2	Teplota v reakčním prostoru	^oC
x_3	Počet zastávek ve výrobě	Měsíc
x_4	Průměrná délka zastávky	Hod
y	Zbytková aktivita katalyzátoru	%

Obr.5.20

Jednotlivé jazykové proměnné mají definované jazykové hodnoty, uvedené v tabulce na Obr.5.21. V této tabulce jsou současně uvedena jména (identifikátory) jazykových hodnot a parametry lichoběžníkových fuzzy množin, které je formalizují. Parametry fuzzy množin jsou dány hodnotami jejich bodů zlomů [*a, b, c, d*].

PROMĚNNÁ	JAZYKOVÁ HODNOTA	JMÉNO JH	*a*	*b*	*c*	*d*
x_1	KRÁTKÝ	PN	0	0	6	15
	STŘEDNÍ	SD	12	18	24	33
	DLOUHÝ	KZ	27	36	48	48
x_2	NÍZKÁ	NIZ	560	560	580	590
	NORMÁL	NOR	585	590	600	610
	VYSOKÁ	VYS	605	615	700	700
x_3	MALÝ	M	0	0	1	3
	STŘEDNÍ	S	2	3	5	6
	VYSOKÝ	E	5	7	20	20
x_4	KRÁTKÁ	KAB	0	0	3	9
	STŘEDNÍ	VYF	100	120	360	480
	DLOUHÁ	VCH	360	480	2300	2300
y	HAVARIJ.	H	0	0	54	60
	NÍZKÁ	N	56	60	66	75
	USPOKOJ.	U	66	75	80	82
	DOBRÁ	D	80	82	87	96
	VÝBORNÝ	V	90	96	100	100

Obr.5.21

ČÍSLO	1	2	3	4	5	ČÍSLO	1	2	3	4	5
1	PN	NIZ	M	KAB	V	**24**	SD	NOR	S	KAB	U
2	PN	NIZ	M	VYF	V	**25**	SD	NOR	E	KAB	N
3	PN	NIZ	M	VCH	V	**26**	SD	VYS	M	KAB	D
4	PN	NIZ	S	KAB	V	**27**	SD	VYS	M	VYF	V
5	PN	NIZ	E	KAB	D	**28**	SD	VYS	M	VCH	U
6	PN	NOR	M	KAB	V	**29**	SD	VYS	S	KAB	U
7	PN	NOR	M	VYF	D	**30**	SD	VYS	E	KAB	N
8	PN	NOR	M	VCH	D	**31**	KZ	NIZ	M	KAB	D
9	PN	NOR	S	KAB	D	**32**	KZ	NIZ	M	VYF	U
10	PN	NOR	E	KAB	U	**33**	KZ	NIZ	M	VCH	U
11	PN	VYS	M	KAB	V	**34**	KZ	NIZ	S	KAB	N
12	PN	VYS	M	VYF	D	**35**	KZ	NIZ	E	KAB	N
13	PN	VYS	M	VCH	D	**36**	KZ	NOR	M	KAB	U
14	PN	VYS	S	KAB	U	**37**	KZ	NOR	M	VYF	N
15	PN	VYS	E	KAB	U	**38**	KZ	NOR	M	VCH	N
16	SD	NIZ	M	KAB	D	**39**	KZ	NOR	S	KAB	N
17	SD	NIZ	M	VYF	D	**40**	KZ	NOR	E	KAB	H
18	SD	NIZ	M	VCH	D	**41**	KZ	VYS	M	KAB	N
19	SD	NIZ	S	KAB	U	**42**	KZ	VYS	M	VYF	H
20	SD	NIZ	E	KAB	U	**43**	KZ	VYS	M	VCH	H
21	SD	NOR	M	KAB	D	**44**	KZ	VYS	S	KAB	H
22	SD	NOR	M	VYF	U	**45**	KZ	VYS	E	KAB	H
23	SD	NOR	M	VCH	U						

Obr.5.22

Báze znalostí je vytvořena 45-ti IF-THEN pravidly. Jejich tvar ukazuje tabulka na Obr.5.22.

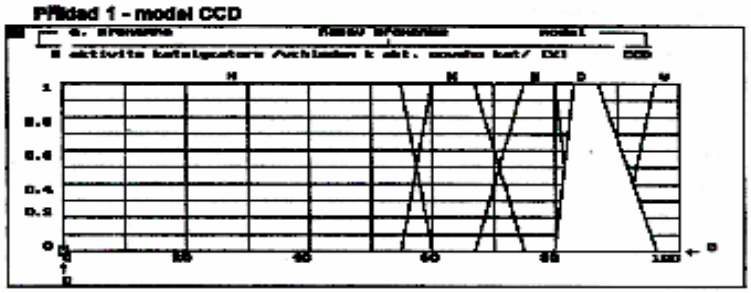
K vytvoření expertního systému je použito prázdného vývojového fuzzy orientovaného pravidlového expertního systému LMPS (Linquistic Model Processing System) [7].

Pro ukázku funkce expertního systému a výsledků vyvozování uvedeme odpovědi fuzzy modelu na tři dotazy, jejichž tvar je uveden na Obr.5.23.

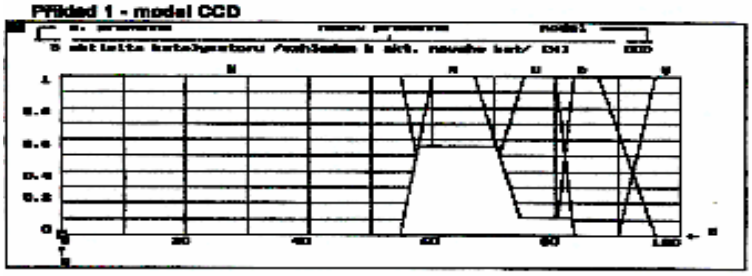
PŘÍKLAD	**TVAR DOTAZU**			
	a	*b*	*c*	*d*
1	20	570	1	850
2	32	595	3	2
3	14	588	1	120

Obr.5.23

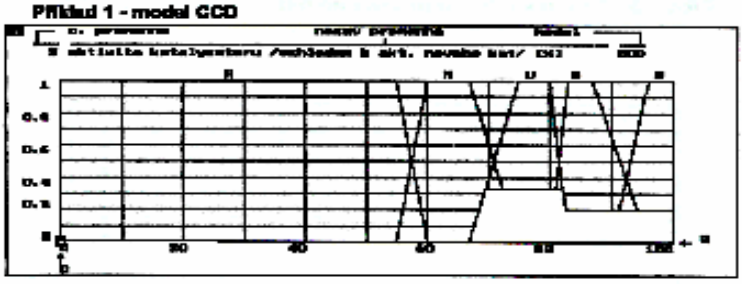
Výsledkem řešení příkladů jsou odpovědi systému, uvedené na Obr.5.24a, Obr.5.24b a Obr.5.24c.



Obr.5.24a



Obr.5.24b



Obr.5.24c

Systém LMPS poskytuje prostředky pro analýzu odpovědí. Do tvaru funkce příslušnosti odpovědi lze vykreslit funkce příslušnosti jednotlivých hodnot závisle proměnné, kurzorem lze odečítat souřadnice významných bodů. Lze vyvolat seznam pravidel, kterých bylo použito pro výpočet odpovědi (aktivní pravidla).

Zájemci si mohou tento programový systém stáhnout z webovské stránky http://homen.vsb.cz/~pok40/.

5.10 Metody defuzifikace

Jak bylo uvedeno v kap.5.7, může být vstupem fuzzy modelu typu Mandami (tedy vstupem expertního systému) ostré obyčejné číslo. Výstupem (odpovědí) takového expertního systému je však vždy fuzzy množina. Obecně nemusí být ani normální, ani konvexní. Interpretace významu takové výstupní fuzzy množiny je diskutována v kap.5.9

V mnohých případech je však nutné, aby výstupem expertního systému bylo ostré číslo. V takové situaci se naskýtá otázka, jakým způsobem nahradit jazykovou interpretaci výstupní fuzzy množiny obyčejným číslem – jedinou numerickou hodnotou z univerza. K tomuto účelu lze použít několik variantních procedur, nazývaných procedury defuzifikace.

Předně zavedeme pojem plochy fuzzy množiny, vyjádřené výrazem

$$\int_U \mu(u) \cdot du \,.$$

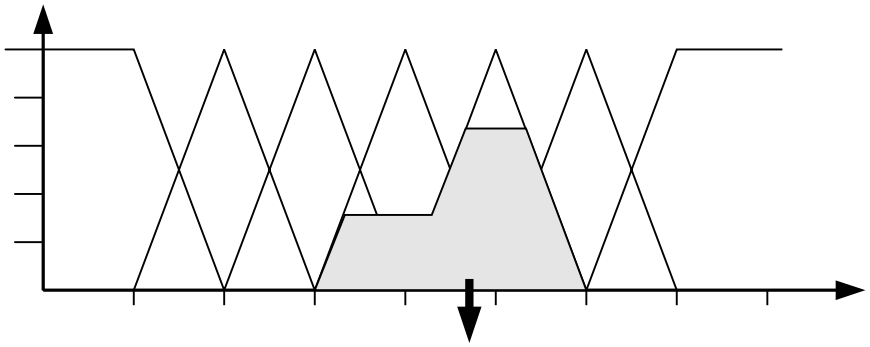
Integrační symbol zde má význam obyčejné integrace (nikoliv význam výčtu jako v předchozím textu). Výšku takové fuzzy množiny budeme značit f_k a hledanou (zastupující) ostrou hodnotu univerza u^*. Uveďme nyní nejčastěji používané metody defuzifikace.

Metoda středu plochy (těžiště) COA

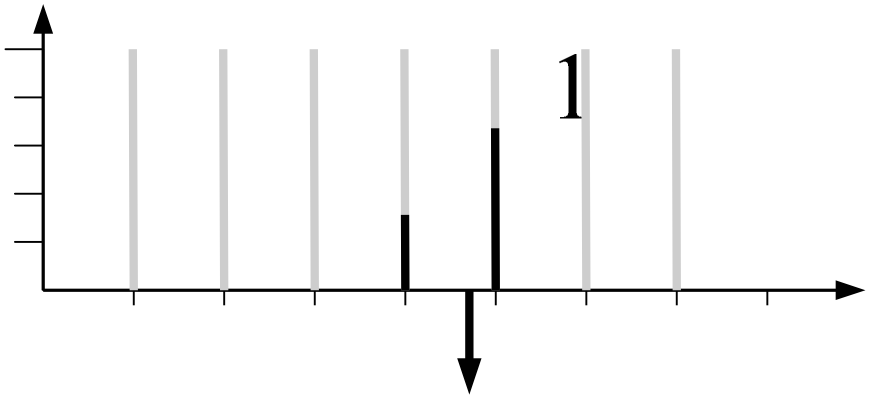
Metoda středu plochy (metoda těžiště) je nejčastěji požívanou metodou defuzifikace. Zastupující ostrá hodnota u^* je určena souřadnicí těžiště plochy výsledné fuzzy množiny. Tato metoda nezohledňuje překrytí (průniky) jednotlivých fuzzy množin výsledného sjednocení (plochy překrytí jsou započteny jen jednou). Pro spojité resp. diskrétní univerzum U platí pro stanovení u^* vztahy

$$u^* = \frac{\int_U u \cdot \mu(u) \cdot du}{\int_U \mu(u) \cdot du} \qquad\qquad u^* = \frac{\sum_{i=1}^{l} u_i \cdot \mu(u_i)}{\sum_{i=1}^{l} \mu(u_i)}$$

kde symbol integrálu označuje obyčejný integrál (plochu). Diskrétní vztah platí i pro funkci příslušnosti typu singleton. Graficky je defuzifikace této metody nakreslena na Obr.5.20 a Obr.5.21.



Obr. 5.20



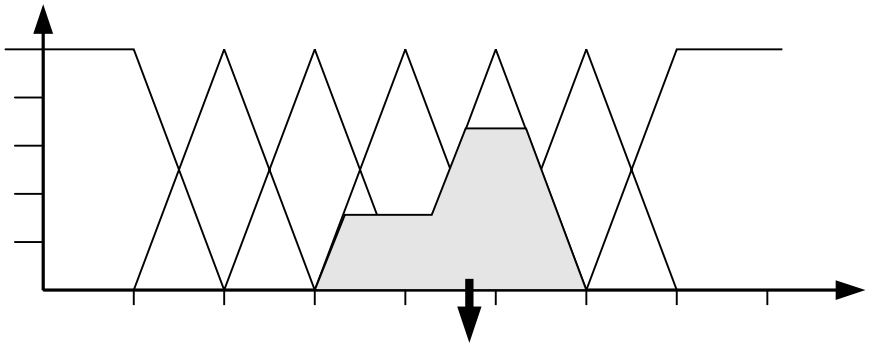
Obr.5.21

Metoda středu součtů COS

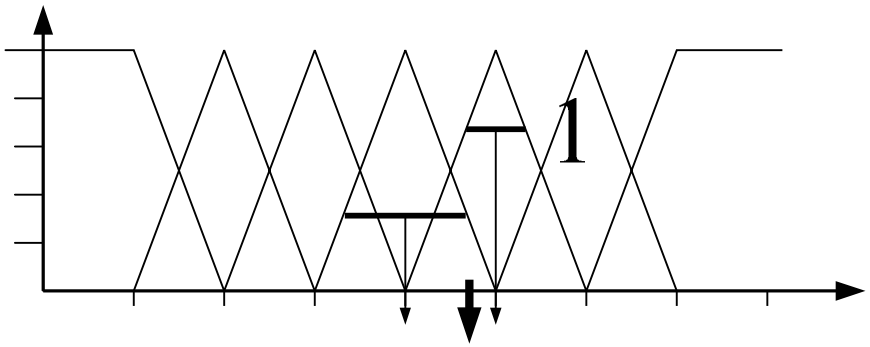
Tato metoda je modifikací metody předešlé a je typická především tím, že zohledňuje průniky jednotlivých funkcí příslušnosti. Nejprve je určen součet funkčních hodnot funkcí příslušnosti ode všech pravidel a teprve pak je určeno těžiště. Pro spojité a diskrétní univerzum U platí vztahy

$$u^* = \frac{\int_U u \cdot \sum_{k=1}^{n} \mu_k(u) \cdot du}{\int_U \sum_{k=1}^{n} \mu_k(u) \cdot du} \qquad u^* = \frac{\sum_{i=1}^{l} u_i \cdot \sum_{k=1}^{n} \mu_k(u_i)}{\sum_{i=1}^{l} \sum_{k=1}^{n} \mu_k(u_i)}$$

kde k je indexová proměnná čísla pravidla a n je počet pravidel. Této metodě odpovídají obrázky Obr.5.22 a Obr.5.23.



Obr. 5.22



NB NM

Obr. 5.23

Výšková metoda HM

Defuzifikace pomocí výškové metody uvažujee špičkové hodnoty každé fuzzy množiny, které jsou váženy výškou fuzzy množiny. Ostrá hodnota je tedy váženým průměrem špičkových hodnot podle vztahu

$$u^* = \frac{\sum_{k=1}^{n} c^{(k)} \cdot f_k}{\sum_{k=1}^{n} f_k}$$

0

kde $c^{(k)}$je špičková hodnota k-té fuzzy množiny a f_k je její výška.

Metoda prvního maxima FOM

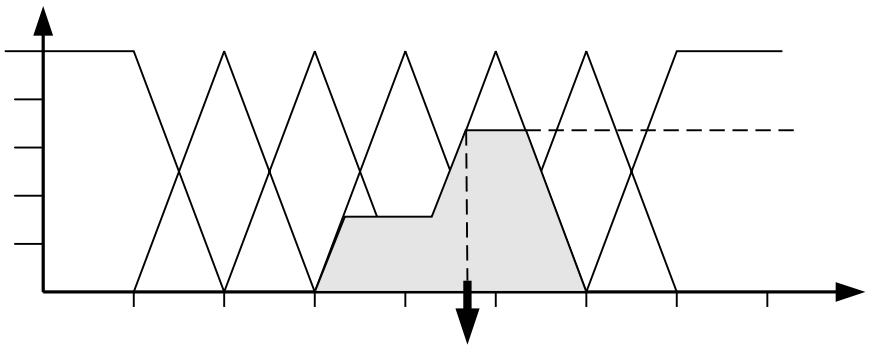
Metoda prvního maxima uvažuje jako ostrou zastupující hodnotu u^* nejmenší hodnotu univerza U, které přísluší maximální stupeň příslušnosti, tj:

$$u^* = \inf_{u \in U_u} \{u \in U_u / \mu_U(u) = hgt(U)\}$$

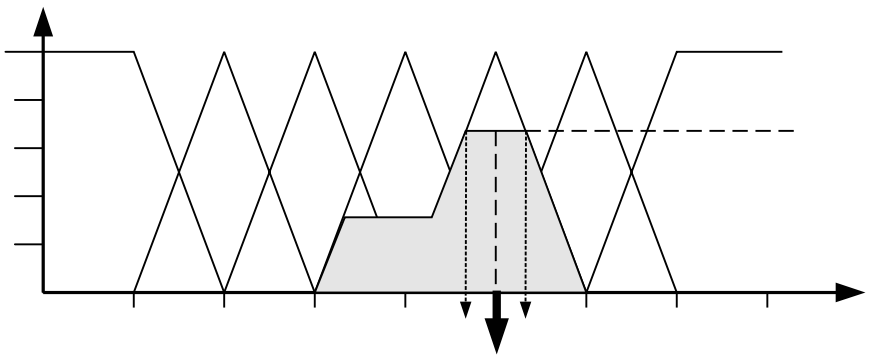
Alternativní verzí je metoda posledního maxima LOM, která jako zastupující ostrou hodnotu uvažuje největší hodnotu univerza U, které přísluší maximální stupeň příslušnosti, tj:

$$u^* = \sup_{u \in U_u} \{u \in U_u / \mu_U(u) = hgt(U)\}$$

Grafické znázornění této metody je na Obr.5.24 a Obr.5.25.



Obr.5.24



Obr.5.25

Metoda středu maxima MOM

Metoda středu maxima je obdobná metodě předchozí, která však jako ostrou zastupující hodnotu bere aritmetický průměr hodnot metody prvního a posledního maxima.

Metody defuzifikace - shrnutí

V případech on-line použití expertních systémů je třeba, aby výstup systému (odpověď) měla formu obyčejného čísla. Kapitola vysvětluje nejpoužívanější metody tzv. defuzifikace. Je to procedura, která transformuje výstupní fuzzy množinu do tvaru zastupující ostré číselné hodnoty. Tato transformace je však provázena vždy ztrátou informace, která fuzzy množina nese.

Kontrolní otázky:

1. Co znamená procedura defuzifikace?
2. Jaké základní metody defuzifikace znáte?

Otázka k zamyšlení:

1. Existují případy expertních systémů, které nevyžadují defuzifikaci výstupní fuzzy množiny?
2. Uveďte příklad expertního systému, kdy je defuzifikace nezbytná.

Úkoly k zamyšlení:

Snižuje se defuzifikací velikost informace, která je nesena odpovědí expertního systému?

Korespondenční úkoly:

Navrhněte, realizujte a odlaďte jednoduchý expertní systém v prostředí LMPS

Shrnutí obsahu kapitoly

Kapitola vysvětluje pragmatické důvody, které vedly ke zobecnění teorie klasických množin. Podává přehled základů teorie fuzzy množin a principů jejich použití pro formalizaci vágnosti jazykových pojmů. Zavádí termín jazykové proměnné a jejích jazykových hodnot. Seznamuje studenty s fuzzy množinovými operacemi, podává vysvětlení definice fuzzy operací t-normy a s-normy, které mají velký význam pro dosažení kvality přibližného usuzování ve fuzzy expertních systémech. Definuje a vysvětluje důležité fuzzy relace, jako cylindrické rozšíření, projekci a kompozici. Uvádí extenzionální princip, který umožňuje rozšíření aritmetiky obyčejných čísel na aritmetiku fuzzy čísel. V rámci pojednání o fuzzy logice zavádí pojem fuzzy výroků a definuje postupy jejich využití ve fuzzy podmíněných *IF-THEN* pravidlech, uvádí

přístupy k interpretaci jak jednoduchých, tak složených fuzzy logických funkcí. Zvláštní pozornost je pak věnována interpretaci složené funkce fuzzy implikace, jsou uvedeny její v praxi nejpoužívanější varianty. Představuje přístup použití zobecnělého pravidla modus ponens pro metodiku stanovení fuzzy množiny jako závěru přibližného usuzování s využitím kompozičního pravidla. Kapitola uvádí množinu fuzzy podmíněných pravidel jako součtový nebo součinový jazykový fuzzy model, který je prostředkem formalizace neurčitých znalostí v bázi znalostí expertního systému. Uvádí přístup kvantitativní parametrizace tvaru fuzzy množiny ve spojitosti s jejím významem jako odpovědi systému a vysvětluje nejpoužívanější metody tzv. defuzifikace jako procedury, která transformuje výstupní fuzzy množinu do tvaru zastupující ostré číselné hodnoty. Pro ujasnění látky uvádí příklady praktických řešení.

Doporučená literatura

[1] Zadeh, L.A.: Fuzzy Sets, Information&Control, 8, 1965
[2] Vysoký,P.: Fuzzy řízení, ČVUT Praha, FEL, 1996
[3] Lukasiewicz,J.: O logice trójwartosciowej, Ruch Filozoficzny, 5, 1920
[4] Jura,P.: Základy fuzzy logiky pro modelování a řízení, VUT Brno, FEI, 1998
[5] Novák,V.: Fuzzy množiny a jejich aplikace, SNTL Praha, 1992
[6] Pokorný, M.: Umělá inteligence v modelování a řízení, BEN Praha, 1996
[7] Drucmůller,M., Rychlý,J.: Linguistic Model Processing System for Personal Computers, VUT Brno, FS, 1988