evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
Slezská univerzita v Opavě
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

# Znalostní a expertní systémy

Studijní opory

**Miroslav Langer**

**Anotace**:

Opory jsou určeny studentům zapsaným v předmětu Znalostní a expertní systémy. Cílem předmětu je seznámit studenty s problematikou reprezentace znalostí, znalostních systémů, fuzzy expertních systémů a znalostního inženýrství.

# Znalostní a expertní systémy

## Studijní opory

**Miroslav Langer**

Slezská univerzita v Opavě
Filozoficko-přírodovědecká fakulta v Opavě
Ústav informatiky

**Poslední aktualizace: 31.7.2012**

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita v Opavě
Bezručovo nám. 13, Opava

Tato inovace předmětu *Znalostní a expertní systémy* je spolufinancována Evropským sociálním fondem a Státním rozpočtem ČR, projekt č. CZ.1.07/2.2.00/28.0014, „Interdisciplinární vzdělávání v ICT s jazykovou kompetencí“.

# Obsah

# 1 Znalostní a expertní systémy

Dříve než se budeme věnovat historii znalostních systémů, uvedeme si několik definic, které znalostní systémy charakterizují.

Znalostní systém je

- Počítačový systém hledající řešení problému v rozsahu určitého souboru tvrzení nebo jistého seskupení znalostí, které byly formulovány experty pro danou specifickou oblast.
- Systém založený na reprezentaci poznatků expertů, které využívá při řešení zadaných úloh.
- Systém kooperujících programů na řešení vymezené třídy úloh, v jednotlivých problémových oblastech obyčejně řešené experty.
- Počítačový systém vybavený znalostmi odborníka (experta) ze specifické oblasti, v rozsahu kterých je schopen učinit rozhodnutí rychlostí a kvalitou vyrovnávající se přinejmenším průměrnému specialistovi.

Každá z definic charakterizuje znalostní systémy z trošku jiného pohledu. V následujícím textu se pokusíme objasnit nejen tyto definice.

# 2 Historie

Dříve, než se začneme bavit o myslících strojích, umělé inteligenci, znalostních a expertních systémech, odpovězme si na otázku, co je to inteligence. Velmi často je nesprávně pojem inteligence spojován s chytrostí. Čtenář jistě nepochybuje o tom, že je inteligentnější, nežli netopýr. Umístěme však čtenáře spolu s netopýrem do stavu beztíže a naprosté tmy. Zcela nepochybně by si to nejeden čtenář s netopýrem vyměnil. Z tohoto jednoduchého příkladu je zřejmé, že na pojem inteligence je třeba pohlížet mnohem komplexněji. Psychologové definují inteligenci různě. My na inteligenci nahlížet jako na schopnost řešit nezvyklé či obtížné situace, schopnost učit se ze zkušeností a přizpůsobit se novým okolnostem. Inteligence je závislá na prostředí.

Vnímání inteligence se s postupem času vyvíjelo. Zatím co ještě počátkem 20. století byla projevem inteligence schopnost hrát šachy, dnes zcela jistě nikdo nepovažuje Deep Blue od IBM, šachový stroj, který porazil Garri Kasparova, úřadujícího šachového velmistra, za inteligentní stroj. Deep Blue není schopen se učit a vítězí hrubou silou na základě vědomostí, které do něj člověk vložil. Je schopen do hloubky několika desítek tahů propočítat všechny možné tahy a zvolit ten nejlepší, což lze těžko považovat za myšlení.

S rozvojem vědy roste i snaha lidstva sestrojit myslící stroje. Již v raném novověku alchymisté spekulují nad vytvořením umělého člověka. Z konce 16. století pochází legenda o pražském golemovi, kterého stvořil rabín Jehuda Löw ben Becalel. V roce 1769 byl hrabě Wolfgang von Kempelen pověřen Marií Terezií sestavením velkolepého stroje. Kempelen sestrojil stroj, který hrál šachy, známý jako Turek. Počátkem 19. století navrhl Charles Babbage analytický stroj. Jeho pokus o sestavení univerzálního stroje však ztroskotal. Na tehdejší dobu byl stroj příliš rozsáhlý a složitý.

S rozvojem výpočetní techniky dochází i ke snaze vytvořit myslící počítače. Vzniká nový obor, umělá inteligence. Počátky umělé inteligence se datují do 50. let 20. století. V roce 1943 publikují Warren S. McCulloch Walter Pitts v „the Bulletin of Mathematical Biophysics 5“ článek „A logical calculus of the ideas immanent in nervous aktivity“, kde popisují mozek jako vzájemně propojené jednoduché buňky. Zavádí zjednodušený model neuronu, dnes označovaný jako McCulloch – Pittsův neuron. Tímto položili základ konekcinostickému

přístupu a umělým neuronovým sítím. V roce 1963 uvádí Allen Newel a Herbert Simon GPS – General Problem Solver. Obecný postup k řešení problémů, který spočívá v dekompozici problému na jednodušší podproblémy, které jsou buďto řešitelné, nebo opět dekomponované.

Počátkem 70 let 20. století se na Stanfordské univerzitě v Kalifornii Feigenbaumův kolektiv zabývá interpolací hmotových spektrogramů vznikajících při analýze struktur složitých molekul neznámých chemických látek. Cílem je odvodit strukturální vzorec zkoumaných neznámých molekul. Spektrální analýza chemických látek je netriviální praktický problém na který jsou všeobecné metody GPS nedostatečné. K počítačovému řešení tohoto problému je nutno implementovat specifické používání znalostí a vědomostí expertem pro řešení konkrétní oblasti. Tyto vlastnosti a vědomosti jsou často velmi těžko zdůvodnitelné, nebo vůbec nezdůvodnitelné. Implementovat znalosti a jejich používání, které expert získal praxí. Implementovat lidskou schopnost efektivně používat specifické znalosti reprezentované samostatnými strukturami údajů. V roce 1971 vznikl jeden z prototypů expertních systémů, Heuristic DENDRAL, později zkráceno jen na DENDRAL. DENDRAL je systém kooperujících programů a údajových struktur. Stal se významnou pomůckou pro odborníky z chemie, který nezřídka překonával očekávání i vlastní schopnosti interpretace spektrogramů. Významem DENDRALU je, že v něm byly aplikovány specifické problémově zaměřené poznatky při řešení problémů počítačem.

Další oblastí, která je velmi přitažlivá pro aplikaci znalostních systémů je medicína. V roce 1976 Edward Shortliffe opět na Stanfordské univerzitě dokončuje projekt MYCIN, který patří mezi nejvýznamnější medicínské znalostní systémy. MYCIN je produkčním expertním systémem a slouží k diagnostice a doporučení léčby krevních infekcí. MYCIN byl posléze rozšířen o diagnostiku dalších nemocí. Dalším systém z oblasti medicíny vznikl v roce 1974 na univerzitě v Pittsburghu. Kolektiv Randolpha Millera a Harry Pople implementovali vědomosti profesora MUDr. Jacka Myerse, šéfa interní medicíny na University of Pittsburgh School of Medicine do systému INTERNIST-I. INTERNIST-I je pravidlový systém pro diagnostiku problémů v interní medicíně. Začátkem 80. Let 20. století vyšlo najevo, že nejcennější částí systémů je báze znalostí, která byla použita u nástupců INTERNISTu CADUCEUSu a Quick Medical Reference (QMR), komerční diagnostické nástroje pro

internisty. Začátkem 80. Let 20. století začíná éra komerčně nasazovaných znalostních systémů.

ES z rozličných oblastí mají společné to, že jejich činnost je založena na heuristických, kvalitativních, zkušenostmi získaných znalostech.

# 3 Poznatky a znalosti

Snahy o pochopení lidské mysli se datuje přinejmenším do doby helénistické. Už Platón či Aristoteles se snažili pochopit a popsat lidské poznání.

Než se začneme věnovat znalostním systémům, objasněme si pojmy *poznatek* a *znalost* a způsoby reprezentace poznatků.

Poznatek je reprodukce určité vymezené části objektivního světa včetně zákonitostí, které v něm platí, je to produkt pracovní, společenské a myšlenkové činnosti lidí, jehož charakteristickou vlastností je *komunikovatelnost*, čili vyjádřitelnost v jisté jazykové podobě.

Znalost definujeme jako vzájemně provázané (měnitelné, doplnitelné) struktury souvisejících poznatků. Znalost něčeho znamená vlastnit tomu odpovídající reprezentaci v podobě dostatečně věrného a přesného kognitivního modelu včetně schopnosti vykonávat s touto reprezentací různé kognitivní operace.

Rozdíl mezi poznatkem a znalostí je velký. Vlastnit soubor poznatků o něčem ještě neznamená tuto oblast znát. Nejeden čtenář si jistě pamatuje na některé své spolužáky, kteří měli, nejčastěji matematiku, chemii či fyziku „našprtanou“, aniž by byli schopni dané poznatky použít. Takový spolužák má zcela jistě o dané problematice spoustu poznatků, kterým však nerozumí, není schopen je aplikovat a tudíž lze těžko říct, že danou problematiku zná.

# 4 Reprezentace poznatků

Už jsme si řekli, že poznatek je reprodukce vymezené části objektivního světa a jeho charakteristickou vlastností je komunikovatelnost. Nejčastěji poznatky a znalosti čerpáme z časopisů, knih, či internetu, zkrátka v „tištěné“ podobě, nebo z přednášek či kurzů v podobě mluveného slova. Lidský mozek je schopen vstřebat psané, či mluvené slovo, případně nákres či rys a tyto vědomosti si uložit a později použít. Jakým způsobem ale tyto poznatky mozek uchovává? Jak jsou poznatky v mozku reprezentovány? Představte si, že se nás turista zeptá na cestu. Nebuďme nevlídní, odpusťme si odpověď typu „promiňte, nejsem odsud“ a snažme se turistovi cestu popsat a soustřeďme se na to, co se děje v našem vlastním mozku, když popisujeme trasu, kterou musí turista urazit, aby se dostal ke svému vytouženému cíli. Mnohým se „před očima“ zobrazí mapa části města, jiní zase tuto trasu „virtuálně projdou“. Psychologové a filozofové se snaží tyto reprezentace popsat. Nám jejich snaha dává poměrně silný nástroj pro reprezentaci a práci s poznatky v IT. Uvedeme si šest základních reprezentací.

## 4.1 Formální logika

Zakladatelem formální logiky byl řecký filozof Aristoteles (386-322 př.n.l.). Z formální logiky vychází řada základních myšlenek o reprezentacích a výpočtu. Aristoteles se systematicky věnoval postupům usuzování. Logika je výkonný nástroj pro výzkum reprezentací i výpočtů. Složitější typy logiky, jako např. *propoziční* a *predikátový kalkul* umožňují vyjádřit mnohé složité druhy znalostí. Mnohé soudy se dají pochopit v pojmech *logických dedukcí* s odvozovacími pravidly typu *modus ponens*, nebo *modus tollens*.

Modus ponens:

p -> q

p

Tudíž q

Modus tollens:

p -> q

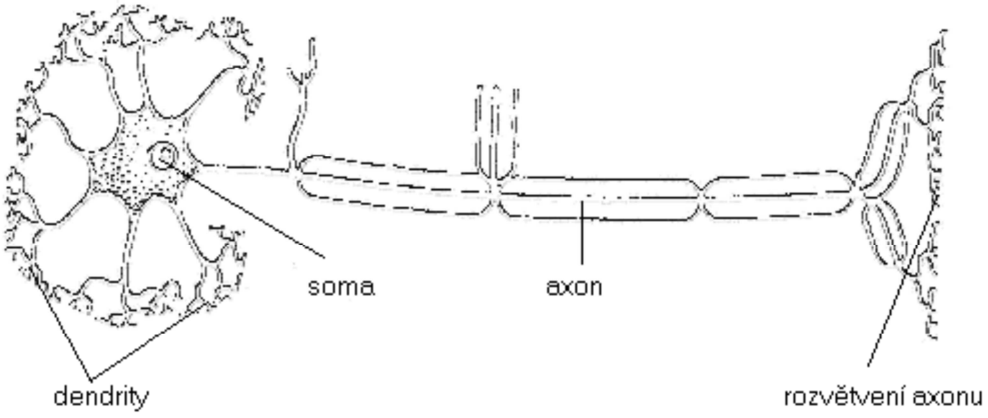
non q

Tudíž non p.

Pravidla predikátové logiky umožňují pracovat s *kvantifikátory*. Pomocí logické dedukce lze řešit celou řadu problémů. Logika například umožňuje plánovat. Nevýhodou je, že i ty nejjednodušší plány potřebují velké množství inferencí a klasická logika nebere v potaz pravděpodobnost a neurčitost.

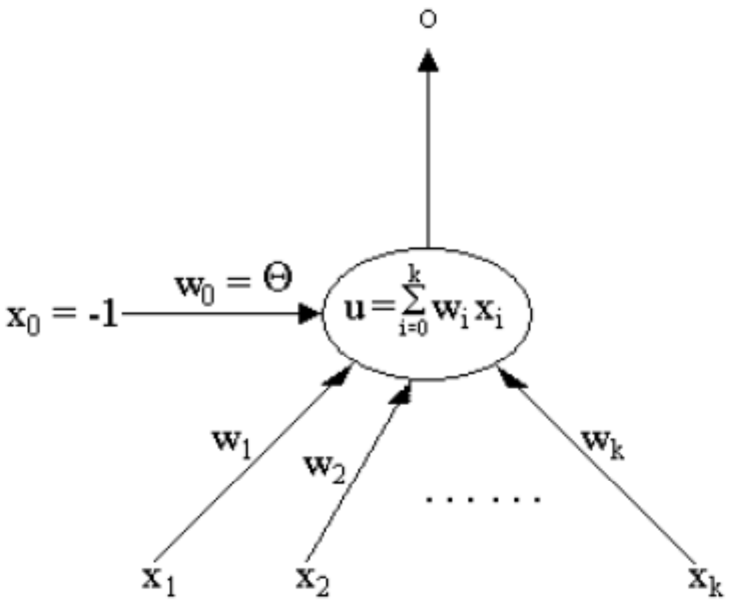
## 4.2 Konekcionistické sítě

Základy konekcionistického přístupu položili McCulloch a Pitts. Konekcionistický přístup k reprezentaci poznatků čerpá inspiraci ze struktury lidského mozku. Spočívá v propojování jednoduchých uzlů.




**Neuron**

*Formální neuron* má *k* obecně reálných vstupních hodnot $x_1 \ldots x_k$, které reprezentují vstupní signály z dendritů. Každý vstup $x_i$ má přiřazenou odpovídající hodnotu *váhy synapse* $w_i$. Tyto váhy reprezentují permeabilitu synaptického spojení. Váhy jsou obecně reálné a mohou nabývat i záporných hodnot, což odpovídá inhibičním vlastnostem synapse. *Práh excitace* neuronu je reprezentován váhou $w_0$ společně se vstupní hodnotou $x_0 = -1$. *Vnitřní potenciál* neuronu je roven sumě součinů vstupních signálů a jim odpovídajících vah $u = \sum_{i=0}^{k} w_i x_i$. *Výstup* z neuronu nebo též výstupní potenciál je dán aktivační funkcí $o = \psi(u)$.




**Formální neuron**

Propojováním těchto jednoduchých umělých neuronů vznikají umělé neuronové sítě. Vědomosti takovýchto sítí jsou reprezentovány váhami neuronů. Umělé neuronové sítě mají schopnost se *učit*. Učení spočívá v adaptaci vah neuronů. Umělé neuronové sítě mají schopnost *predikce, klasifikace, aproximace*.

## 4.3 Představy

Jak sám pojem napovídá, představy jsou obrazové reprezentace vizuální a prostorové informace. Obrazová reprezentace je v některých případech mnohem použitelnější než opis, pro, například CAD systémy zcela nezbytná. Těžko si představit, kterak projektant slovně popisuje projekt rodinného domu. S obrazy lze provádět operace jako je prohlížení, vyhledávání, rotace, transformace. Jsou ovšem znalosti, které pomocí představ vyjádřit nelze, jako například abstraktní výroky, nebo obecně platné věty.

## 4.4 Analogie

Značnou část problémů řešíme pomocí analogií. Uvažování v analogiích je uvažovaní založené na příkladech. Zvládání nových situací založené na přizpůsobení si situace podobné, kterou již známe. Analogie je vhodné použít tam, kde nemáme k řešení problému k dispozici

pojmosloví a neznáme pravidla. Analogie se používají při rozhodování. Výběr alternativ je založen na předešlých případech – *causy*. Tyto případy pak slouží jako *precedenty*. Precedentní právo je například v USA. Učení je založené 1) na ukládání všech souvisejících případů na předchozí zkušenosti do paměti, 2) přizpůsobení předchozího případu k řešení nové situace, 3) zobecnění předešlých analogů a vytvoření analogického schéma.

## 4.5 Pojmy – koncepty

Pojmy se zabýval už Platón. Pojem je charakterizován souborem typických vlastností. Například pojem „fretka“ lze charakterizovat jako obratlovce, čtyřnohého suchozemského masožravého savce, tchořovitou šelmu. To, co je pro fretku typické představuje schéma pojmu fretka. Z této charakteristiky lze vyčíst i základní vlastnosti pojmů. Možnost hierarchizace a dědičnost. Každá fretka je savec, ale ne každý savec je fretka. Pojem savec je nadřazený pojmu fretka a naopak pojem fretka je podřazený pojmu savec. Každá fretka dědí vlastnosti savců. Pojmy jsou vhodným doplňkem pravidel.

## 4.6 Pravidla

Pravidla neboli struktury *jestliže-pak* se skládají ze dvou částí. 1. část je předpoklad, podmínka, neboli *antecedent*, 2. část tvoří důsledek, *konsekvent*. Pokud je podmínka splněna, pak platí důsledek. Podmínková část může obsahovat i podmínky složené, čili spojené logickými spojkami AND nebo OR. Důsledková část může obsahovat spojky AND.

Pravidla nás provází na každém kroku. Pokaždé, když chceme přejít silnici, použijeme několik pravidel:

1. JESTLIŽE chci přejít silnici PAK se musím rozhlédnout na obě strany
2. JESTLIŽE jede auto PAK musím počkat až přejede
3. JESTLIŽE nejede žádné auto PAK můžu přejít

Čtenář jistě cítí možnost úpravu těchto pravidel s ohledem na vzdálenost a rychlost jedoucího auta ve vztahu k možnosti přecházení. Jako další příklad z běžného života můžeme uvést pravidlo JESTLIŽE je venku pod nulou PAK zamrzne rybník. I toto pravidlo lze doplnit o další údaje, jako je např. závislost teploty na rychlosti zamrzání rybníka. Na příkladech je vidět, že pomocí pravidel můžeme reprezentovat informaci o světě a informaci jak se chovat

ve světě. Dále mohou pravidla reprezentovat vyvozovací pravidla typu modus ponens nebo modus tollens. Pravidla mohou utvářet i rozvité struktury.

Řešení problémů pomocí pravidel spočívá v prohledávání prostoru možností – *stavového prostoru*. Pro složitější problémy je charakteristické velké množství možností, čili velmi rozsáhlý stavový prostor. V těchto případech je mnohdy nemožné projít všechny možnosti a nalézt řešení. Takovéto stavové prostory prohledáváme s využitím heuristik. K prohledávání stavového prostoru používáme dva způsoby, *zpětné* a *dopředné* prohledávání stavového prostoru. Při zpětném prohledávání stavového prostoru, neboli *cílem řízeném* odvozování máme předem stanovený cíl a hledáme předpoklady potřebné pro dosažení tohoto cíle. Slouží k potvrzení nebo vyvrácení hypotézy, či zodpovězení dané otázky. Při dopředném prohledávání stavového prostoru, neboli *fakty (údaji) řízeném* odvození určujeme řešení, které vyplývá z určitého objemu známých dat (faktů, údajů), které je zapotřebí interpretovat. Stanovujeme, co z daných faktů vyplývá a k jakým důsledkům je z nich možno dospět. Při dopředném odvozování vycházíme z daných údajů, které se v průběhu interpretace mohou doplňovat.

# 5 Determinismus a nedeterminismus

Při strojovém řešení problémů klasickým imperativním programováním je nutné, aby byl řešený problém algoritmizovatelný, čili řešitelný deterministicky. V každém kroku řešení problému je předem dané, jaké kroky budou následovat. Jedná se o tzv. reproduktivní řešící postupy.

Oproti tomu v případě nedeterministických, produktivních řešících postupů problémů není zřejmé, jaký krok bude následovat v daném kroku. Výběr dalšího kroku řešení je závislý na výsledku testování podmínky, o které nelze s určitostí říct, zda je či není splněná. Ze samotné formulace problému ani z průběhu jeho řešení není zřejmý další krok řešení. V Jednotlivých krocích řešení můžeme volit z množiny kroků, které budou následovat. Volba následného kroku je tak závislá na hádání, odhadování, volbě či jiném nedeterministickém výběru jednoho z alternativních možností řešení. Čtenář si jistě vzpomene na definici nedeterministického Turingova stroje a jistě si i pamatuje, že nedeterministický TS přijímá slovo, pokud existuje alespoň jedna posloupnost konfigurací končící v koncovém / akceptačním stavu. Formulace „alespoň jedna posloupnost“ říká, že při řešení můžeme dojít do slepé cesty, můžeme se dostat na konfliktu, může existovat jiný, lepší, kratší postup řešení. Volba následujícího kroku tedy není závazná. Můžeme po revizi řešení od současného řešení odstoupit od současného řešení a pokračovat v jiné vhodnější variantě.

# 6 Znalostní systémy

Jedna z definic říká, že znalostní systém je počítač vybavený programy umožňujícími řešit problémy na základě poznatků produktivními postupy - systém vybavené jistou znalostí.

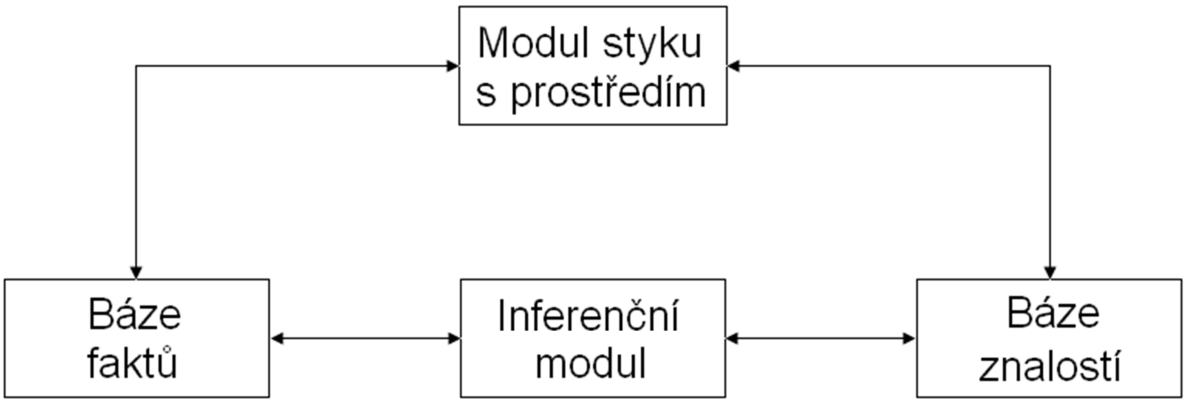
Znalostní systémy jsou tedy vybaveny znalostmi a jejich úkolem je řešit netriviální problémy z různých oblastí lidské činnosti. Znalostní systémy jsou nasazovány tam, kde potřeba znalostí experta a z různých důvodů nelze využít jeho služeb. Slouží k řešení problémů, které nejsou jednoduše, nebo nejsou vůbec algoritmizovatelné. Důvody k nasazení systému na místo člověka mohou být například ekonomické. Zavedení znalostního systému je finančně výhodnější a dostačující, než placení experta. Podpůrné rozhodovací systémy (DSS – Decision Support Systems) mohou být nápomocny méně zkušeným pracovníkům, nebo pracovníkům s nižším vzděláním v dané oblasti. DSS mohou nebo poskytovat alternativy pro experty. V neposlední řadě mohou expertní systémy sloužit k výuce (viz. INTERNIST-I). Znalostní systémy jsou nasazovány i v prostředích, kde je třeba okamžitého rozhodnutí v řádu vteřin i méně. V prostředích, kde je třeba zpracovat stovky, či tisíce údajů, provádět složité výpočty a reagovat v časech, které jsou pro člověka nedosažitelné.

Znalosti ve znalostních systémech jsou reprezentovány ve formě symbolických výrazů odpovídající deskripčním a relačním zákonitostem. Rozsah takto reprezentovaných znalostí je omezen konceptuálním systémem pro popis dané problematiky a formou jeho symbolické reprezentace zpracovatelnou počítačem. Volba formalismu značně ovlivňuje výpočetní schopnosti systému.

Znalostní systém je soustava kooperujících programů. Funkčně vymezený programový celek se nazývá modul. Jednotlivé moduly jsou (ne nutně vzájemně) provázány. Moduly a jejich vazby tvoří architekturu znalostního systému.

Základní architekturu znalostních systémů tvoří čtyři moduly, jejichž funkci popíšeme v dalších přednáškách:

- Báze znalostí (BZ)
- Báze faktů (BF)
- Inferenční modul (mechanismus) (IM)
- Modul pro styk s prostředím




**Architektura znalostního systému**

# 7 Expertní systémy

Pozorný čtenář si jistě povšiml, že se v přednáškách vyskytují dva pojmy pro zdánlivě též systémy, znalostní a expertní systémy. Byť v poslední době oba pojmy splývají, my budeme oba pojmy definovat samostatně.

Expertní systém definujeme jako počítačový systém hledající řešení problému v rozsahu určitého souboru tvrzení nebo jistého seskupení znalostí, které byly formulovány experty pro danou specifickou oblast, systém založený na reprezentaci poznatků expertů, které využívá při řešení zadaných úloh, systém kooperujících programů na řešení vymezené třídy úloh, v jednotlivých problémových oblastech obyčejně řešené experty počítačový systém vybavený znalostmi odborníka (experta) ze specifické oblasti, v rozsahu kterých je schopen učinit rozhodnutí rychlostí a kvalitou vyrovnávající se přinejmenším průměrnému specialistovi. Většina těchto specifikací je shodná či velmi podobná specifikacím znalostních systémů. Expertní systémy jsou podtřídou znalostních systémů. Vytváří se pro řešení problémů, kterými se obvykle zabývají odborníci. Na rozdíl od znalostních systémů jsou schopny na základě znalostí vysvětlit a zdůvodnit postupy řešení. Tato velmi významná vlastnost umožňuje uživateli posoudit úroveň expertízy, řešení předložené systémem přijmou, modifikovat nebo odmítnout. Taktéž umožňuje uživateli odpovědět na otázky typu, proč, jak, co kdyby, co když ne, atp.

Základním požadavkem strojového zpracování dat a řešení problémů je deterministický postup řešení. Kde se používá „selský rozum“ počítač selhává. „Selský rozum“ lze charakterizovat jako selektivní využívání všeobecných znalostí. Počítač je mnohem jednodušší naučit využívat efektivní uspořádání systému poznatků expertů. Experti mají své poznatky velmi dobře utříděné a strukturované, na rozdíl od „selského rozumu“, kde není taková konceptualizace k dispozici.

Problémy vhodné na řešení expertními systémy patří podle (HAYES-ROTH, WATERMAN, LENAT, 1983) aspoň do jedné z následujících kategorií:

- Interpretace
    - Rozpoznání situace na základě údajů, které ji opisují
- Predikce
    - Odvození očekávatelných důsledků dané situace

- Diagnostika
    - Určení stavu (poruchy, poškození) systému na základě pozorovatelných (dostupných) projevů jeho chybové chování
- Projektování
    - Výběr a sestavení objektů do určitého funkčního celku na základě ohraničujících podmínek
- Plánování
    - Sestavení posloupnosti akcí za účelem dosáhnutí daného cíle
- Monitoring
    - Sledování a porovnávání údajů odpovídajících určité situaci za účelem zjišťování (a následného odstraňování) odchylek od očekávané situace
- Ladění a korekce
    - Výběr, sestavení a uskutečnění posloupnosti akcí odstraňujících odchylky nebo chybové stavy
- Učení
    - Diagnostika, ladění a upravování vědomostí studenta
- Řízení
    - Interpretace, predikce, monitoring a opravování chování systému

Uvedené kategorie problémů lze rozdělit do dvou skupin analyzující a syntetizující. Analyzující problémy jsou takové, jejichž řešení spočívá v rozpoznání, určení předem specifikované, a tedy již popsané entity na základě dostupných údajů. Typickým příkladem je např. identifikace chemické látky, nebo diagnostika. Syntetizující problémy jsou takové, jejichž řešením je, na základě daných údajů a ohraničujících podmínek sestrojení (odvození) zatím ještě neopsané (neznámé) entity známých prvků. Typickým příkladem je konstruování nebo plánování.

Odpověď na otázku, kdy je vhodné použít expertní systémy, spočívá v charakterizaci problému, řešení, principu řešení a požívaných údajů:

- Problém
    - Není formálně dobře vyjádřitelný nebo strukturovatelný

- Řešení
    - Není deterministické, je založené na produktivních postupech vyhledávání vhodné posloupnosti kroků řešení
- Princip řešení
    - Nemá teoreticky dobré a ucelené podklady, použité znalosti nejsou kategorické a formálně dobře vyjádřitelné
- Používané údaje
    - Jsou vágní, nepřesné, nespolehlivé a vzhledem na nedostupnost neúplné

# 8 Základní moduly znalostních systémů

V předešlé přednášce jsme věnovali pozornost architektuře znalostních systémů. Uvedli jsme čtyři základní moduly. V následujícím textu se budeme podrobně věnovat bázi znalostí a bázi faktů a okrajově popíšeme inferenční mechanismus, kterému se budeme věnovat v samostatné přednášce. Tyto tři moduly tvoří jádro znalostních systémů. Inferenční mechanismus na základě dat z báze faktů a znalostí z báze znalostí hledá řešení zadaného problému. Zřejmou podmínkou, aby tyto tři moduly mohli spolu komunikovat je, aby pracovaly se stejnou reprezentací poznatků.

## 8.1 Báze znalostí (BZ)

Báze znalostí (poznatků), jak sám název napovídá, je modu obsahující znalosti nezbytné pro řešení problémů z oblasti, pro kterou byl daný znalostní systém vytvořen.

Báze znalostí je nedílným komplementem inferenčního mechanismu. Lze ji chápat jako deklarativní program, jejíž obsah je, až na výjimky, tvořen pasivními údajovými strukturami. Tyto struktury slouží k parametrizaci aktivních procesů IM, tj. k inicializaci a ovlivňování činnosti IM. V bázi znalostí se, až na výjimky, neprovádí žádné výpočty, netvoří ji vykonatelné instrukce programu. Mohou se v ní však vyskytovat i procedurálně reprezentované poznatky, čili poznatky tvořené vykonavatelnými instrukcemi programovacího jazyka, které ovšem zabezpečují pouze specifické lokální výpočty, jejímž výsledkem není opět nic jiného, než parametry všeobecných procesů inferenčního mechanismu.

Báze znalostí se v průběhu inference nemění, její obsah zůstává *invariantní*. Bázi znalostí je ovšem možno měnit či rozšiřovat, pokud se v ní odhalí nedostatky či chyby. Báze znalostí některých systémů disponují schopností se učit.

Pro danou oblast použití znalostního systému je nutné zvolit vhodnou reprezentaci poznatků. Při volbě reprezentace je nutno brát v potaz

- *vyjadřovací schopnosti*, tj. co a do jaké míry je jí možné reprezentovat,
- *odvozovací schopnosti*, tj. co a jak umožňují odvozovat

- *výpočetní schopnosti* (efektivitě), tj. o složitosti symbolových procedur, které jsou jí podmiňované.

Volba reprezentace by měla úzce souviset s analogickými úvahami o inferenčním mechanizmu pracujícím s bází znalostí.

Při uvažování volbě reprezentace je vhodné mít na zřeteli různorodou povahu odborných poznatků a možnost jejich třídění (kategorizace). Experti často disponují následujícími typy poznatků:

- *asociativní* - odpovídají obvyklým, pozorovatelným nebo pravděpodobným, ne však zdůvodnitelným a ani nevyhnutelným vztahem mezi dvojicemi entit (např. jejich současný nebo následný výskyt)
- *kauzální* – odpovídají známé a zdůvodnitelné souvislosti mezi dvěma entitami, které jsou anebo můžou být ve vztahu příčiny a následku.
- *taxonomické* – odpovídají známým souvislostem mezi entitami, které vystihují vztah všeobecnému k speciálnímu (generalizačně-specializované vztahy)
- *kontextové* – odpovídají různým podmínkám, za kterých se uplatňují určité souvislosti mezi entitami (vyjadřují např. okolnosti, na kterých závisí, jestli se souvislost uplatní nebo neuplatní)
- *prostorové* – odpovídají prostorovým vztahům mezi entitami (např. vztahem sousedů, části k celku, polohy uvnitř nebo vně, polohy a orientace prostoru, směr pohybu atd.)
- *časové* – odpovídají takovým souvislostem mezi entitami, které na základě jejich výskytu anebo jejich změn v určitých časových okamžicích nebo intervalech podmiňujících okamžité anebo časově následující výskyty nebo změny jiných entit a tím umožňují uvažovat o jevech minulých a budoucích.
- *modelové* – odpovídají známým, nebo předpokládaným dynamickým souvislostem zpravidla mezi větším počtem určitým způsobem provázaných entit, které se vzájemně ovlivňují a tím určují chování se jimi vytvořeného systému.

Existují znalostní systémy, jejichž báze znalostí je prázdná. Takové systémy nazýváme *prázdné* (shell) znalostní/expertní systémy. Takové systémy disponují inferenčním mechanismem s univerzálními řešícími postupy. Pro jeden inferenční mechanismus je možno vytvořit několik různých bází znalostí. Jednotlivé báze znalostí se mohou lišit jak obsahem, tak použitím a systém se tak stává použitelným i pro značně problémově vzdálené oblasti. Při

naplňování báze znalostí je třeba brát v potaz reprezentaci poznatků, se kterou pracuje inferenční mechanismus prázdného systému. Efektivita prázdného znalostního systému, pro který vytvoříme bázi znalostí, jistě nebude srovnatelná se systémem speciálně vytvořeným pro danou oblast. Proto je vhodné zvážit, jestli budou schopnosti takového systému dostatečné. Naplnění báze znalostí prázdného systému je vhodné použít pro tvorbu relativně jednoduchých systémů. Pro řešení problémů musí být dostačující jednoduché řešící postupy, případně univerzální heuristiky.

## 8.2 Báze faktů (BF)

Báze faktů (údajů) je druhý pasivní komplement inferenčního mechanismu. Obsahuje symbolovou reprezentaci všeobecně platných poznatků z dané problémové oblasti, dále uchovává konkrétní fakta související s právě řešeným problémem. Na rozdíl od báze znalostí, se báze faktů v průběhu inference může značně měnit. Inferenční mechanismus může v průběhu inference doplňovat, měnit nebo rušit jednotlivá fakta. Báze faktů je přístupná i dalším aktivním modulům znalostního systému, o kterých ještě bude řeč.

Způsob organizace báze faktů a tvar položek musí odpovídat formalismu BZ a funkcím IM. Báze faktů může být tvořena například symboly entit s přiřazenou hodnotou, seznamem entit se společnými vlastnostmi, n rozměrnými vektory, případně jiné složitější struktury.

Frekvence přístupu do BF je vyšší, než v případě BZ, což vyžaduje umožnění snadného přístupu k položkám. Časová složitost prohledávání struktury BF musí být co nejnižší. Z těchto požadavků vyplývá nutnost velmi dobré organizace obsahu BF, položky musí být vhodně strukturované a naformátované.

Na začátku řešení problému obsahuje báze faktů *inicializační položky*, které vznikají při počáteční specifikaci problému a jsou obecně platné pro daný problém. Položky BF mohou být doplňovány v průběhu řešení na základě stávajících faktů a aplikovaných znalostí na tato fakta. Jedná se například o důsledkové části pravidel. Položky mohou vznikat odhadem. Takové položky můžeme označit jako očekávané, nebo předpokládané. Pravdivost takto doplněných položek je nutné následně ověřit a jsou platné, dokud nejsou ve sporu s realitou. Spolu s požadavkem na rychlé a efektivní prohledávání báze faktů je spojena možnost

odstranění redundantních faktů. Některá fakta jsou *komplementární*, tj. z jednoho faktu vyplývá druhý (kočka není pes, atp.). Dále může vznikat redundance na základě hierarchie, například nepotřebujeme držet informaci o tom, že pes je savec, nebo naopak nám stačí informace, že daná entita je savec a je nám lhostejné, jestli je to delfín, nebo slon.

## 8.3 Základní pohled na inferenční mechanismus (IM)

Inferenční mechanismus tvoří jádro procedurálních složek znalostních systémů. Prohledáváním stavového prostoru vyhledává řešení daného problému.

Složitější problémy charakterizuje velké množství možností, čili rozsáhlý stavový prostor. Projít všechna možná řešení je v takovýchto případech časově velmi náročné, ne-li nemožné, proto je nezbytně nutné používat heuristiky.

K prohledávání stavového prostoru používáme dva způsoby, zpětné a dopředné prohledávání. Zpětné prohledávání stavového prostoru, neboli cílem řízené odvozování slouží k potvrzení nebo vyvrácení hypotézy zodpovězení dané otázky, odvození způsobu konstrukce daného zařízení, zjištění předpokladů (požadavků), dosáhnutelnost daného cíle. Při zpětném prohledávání stavového prostoru máme předem stanovený cíl a hledáme předpoklady potřebné pro jeho dosažení. Pokud není možné bezprostředně rozhodnout o splnitelnosti všech podmínek, pak hledáme alternativní podmínky splnitelnosti, případně určíme nové (pod)cíle řešení a rekurzivně aplikujeme postup na nové (pod)cíle, dokud není splnitelnost předpokladů bezprostředně ověřitelná na základě dostupných faktů.

Při dopředném prohledávání stavového prostoru určujeme řešení, které vyplývá z určitého objemu známých dat (faktů, údajů), které je zapotřebí interpretovat, čili je třeba stanovit, co z nich vyplývá a k jakým důsledkům je z nich možno dospět, přičemž vycházíme z daných údajů, které se v průběhu interpretace doplňují. Počáteční (iniciální) údaje, které charakterizují počáteční stav řešeného problému, však jen zřídka postačují k jeho vyřešení. Fakta je nezbytné postupně doplňovat. Pokud z počátečních údajů explicitně žádný cílový stav nevyplývá, nemůže z nich ani explicitně vyplývat to, jaká fakta, v jakém počtu a pořadí je třeba dále zkoumat. V takovém případě je však možné, na základě reprezentovaných znalostí o objektivním světě, se, pomocí asociací či heuristik, zaměřit na nějakou hypotézu a tu následně ověřit.

# 9 Inferenční mechanismus

Inferenční mechanismus tvoří jádro procedurálních složek znalostního systému. Simuluje uvažování experta.

Na inferenční mechanismus lze nahlížet ze tří základních pohledů:

- vnější, uživatelský pohled – jako podstatné se jeví ty funkce IM, které vnímá uživatel. Chování těchto funkcí označujeme termíny psychologie a logiky, jako např. deduktivní, induktivní uvažování, asociativní usuzování, kauzality, generalizace, analogie, heuristické usuzování… Z pohledu uživatele je IM prostředek, který realizuje schopnost uvažování experta.
- Vnitřní pohled – zaměřujeme se na funkce spojené s teoretickými principy konstrukce IM. Tyto funkce nejlépe vystihují pojmy matematické informatiky, jako např. teorie grafů, konceptuální grafy, matematická logika, matematická lingvistika, teorie složitosti, časových řad, teorie množin, teorie pravděpodobnosti…
- Implementační pohled – funkce vnímáme z pohledu programovacích jazyků a programového prostředí. Zajímají nás vlastnosti operačního systému, na kterém se systém bude provozovat, knihovny programů, které máme k dispozici, možnosti řízení BF…

Pouhé omezení se na zpětné a dopředné řetězení při tvorbě IM není pro aplikaci systému v praxi většinou dostačující. Při navrhování inferenčního mechanizmu je třeba brát v potaz kognitivní schopnosti expertů:

- Globální pohled na řešený problém – schopnost vnímat problém nejen na základě lokálních požadavků, ale pohlížet na problém jako celek z globálních pohledů, což umožňuje nacházet optimální řešící postupy, které nejsou z lokálních pohledů zřejmé.
- Vytvářet víceúrovňové abstrakce řešeného problému – schopnost hierarchizování řešeného problému umožňuje zacházet do podrobností pouze tehdy, kdy je to účelné, rozpoznávat podstatné a podružné informace.
- Dekompozice problému – schopnost zjistit, kdy je vhodné rozložit problém na podproblémy, které jsou samostatně řešitelné. Pokud se některé podproblémy vzájemně ovlivňují, pak volbou nezávazných parametrů interakce eliminovat.

- Doplňovat chybějící údaje předpokládanými – schopnost nezávazně doplnit chybějící údaje, které se následně buď potvrdí, nebo vyvrátí.
- Odhalovat souvislosti mezi fakty - umožňuje dynamické zaměření pozornosti na významné charakteristiky problému
- Transformace problému na analogický nebo přeformulovat problém na vhodnější tvar, řešení takto transformovaného problému je lehčí.
- Generalizace poznatků získaných z úspěšného, ale i neúspěšného řešení problému, které jsou použitelné pro řešení problémů dalších.

Při návrhu efektivního IM je třeba brát v potaz i další schopnosti expertů. V různých oblastech aplikace ZS se požadavky na IM různí. Problémy úspěšně řešitelné pomocí základních prostředků IM jsou vesměs problémy vznikající v úzkých problémových oblastech a jsou charakterizovány:

- malým stavovým prostorem řešení,
- ustálenými řešícími postupy, které nevyžadují hierarchizování,
- nenáročnými požadavky na strukturu a organizaci potřebných znalostí
- dostupností faktů nezbytných k řešení.

Takové problémy lze poměrně efektivně řešit pomocí zpětného a dopředného řetězení. Většina reálných problémů však tyto charakteristiky, bohužel, nesplňují. Pro většinu reálných problémů je naopak charakteristické:

- rozsáhlý stavový prostor, který je nutné, nebo účelné hierarchizovat nebo členit,
- je nutné použít rozsáhlé BZ obsahující typově různé poznatky,
- některá fakta jsou neúplná, nejednoznačná, nebo nespolehlivá,
- je nutné je dekomponovat na podproblémy, ve kterých často nejsme schopni určit veškeré parametry řešení, což sebou přináší odhadování potřebných faktů, navíc řešení některých podproblémů může být vzájemně podmíněné.

Při vytváření IM, a pochopitelně i při návrhu reprezentace poznatků, je nutno brát tyto a podobné charakteristiky reálných problémů v potaz.

Hlavním požadavkem na ZS je schopnost úspěšně a spolehlivě řešit problémy, požadujeme *úplnost* odvozovacích procesů, tj. pokud řešení daného problému existuje, pak jej ZS nalezne. Samotná úplnost není dostačující. Důležitým kritériem je i *způsob a efektivnost odvozování*, tj.

jeho kvalita. Triviálním požadavkem na kvalitu řešících postupů je, aby se řešící postupy neopakovaly, byly *neredundantní*. Kvalita je dána i rozsahem a způsobilostí IM realizovat *kognitivní schopnosti*, které charakterizují experta.

## 9.1 Řízení inference

Z požadavků na kvalitu odvozování vyplývá potřeba hlubší analýzy odvozovacích procedur.

Ačkoliv jsme si uvedli několik způsobů reprezentace poznatků, lze, v konečném důsledku všechny zmiňované, snad s výjimkou konekcionistických sítí a představ, reprezentovat pomocí struktur IF – THEN. Poznatek reprezentovaný např. rámcem není nic jiného, než vyhodnocení, vesměs velkého množství, předpokladů. Totéž platí pro asociativní sítě či logiku. V případě konekcionistického přístupu jsou však řešící postupy poněkud odlišné. Představy zase často vstupují do systému, jako zadání problému, např. rekognice obličeje, RZ automobilu, atp.

Bez ohledu na zvolenou formální reprezentaci spočívá jádro odvozování na vyhodnocování vesměs velkého počtu různých podmínek a na zohledňování důsledků z nich vyplívajících. Na základě toho se omezíme pouze na formalismus produkčních pravidel.

Při vyhodnocování konkrétního pravidla se můžeme dostat do situace, že všechny podmínky předpokladové části pravidla nejsou známy, nejsou obsaženy v bázi faktů. Jednou z možností, jak tyto podmínky doplnit je odhadnout je. I tyto odhadnuté podmínky se stávají součástí BF. Bázi faktů tak můžeme rozdělit na tři, navzájem disjunktní podmnožiny:

- fakta odpovídající tvrzení TRUE {T},
- fakta neodpovídající tvrzení FALSE {F},
- nerozhodnutelné, neznámé UNKNOWN (UNDECIDEABLE) {U}, čili fakta, o jejichž pravdivostní hodnotě v daném okamžiku nejsme schopni rozhodnout.

BF je tvořena konjunkcí těchto množin.

V průběhu inference se každé pravidlo nachází v jednom ze tří stavů:

- neuvažované {N} – doposud nevznikl v průběhu řešení stav, který by vyžadoval jeho použití,
- aktivní {A} – vznikl požadavek na užití tohoto pravidla, ale doposud jeho interpretace nebyla uskutečněná, nebo ukončená,
- deaktivované {D} – intepretace byla dokončena.

K ukončení intepretace pravidla dojde tehdy, pokud je možno jeho předpokladové části přiřadit pravdivostní hodnotu a tím je možné interpretovat jeho důsledkovou část. Po deaktivaci pravidla se jeho důsledková část stává prvkem jedné z podmnožin {T}, {F}, {U}.

Stav daného problému ve stavovém prostoru je definován:

- stavy jednotlivých pravidel z BZ,
- obsahem báze faktů,
- obsahem množiny faktů, jenž mají být odvozeny, požadovaných faktů {P}

Množinu {P} tvoří fakta, která mají být v průběhu inference vyhodnocena. Jsou to důsledky pravidel, jejichž vyhodnocení je řešením daného problému. Může se však jednat i o fakta, která jsou potřebná pro předpokladovou část pravidel.

Inferenční mechanismus k intepretaci pravidel využívá dvou základních procedur:

- Procedura přímého chodu, neboli dopředná, fakty řízená inference
    - na základě obsahu BF a BZ se z BZ vybereme pravidlo, které se pokusíme interpretovat. Pokoušíme se z dostupných faktů získat důsledky pravidel.
- Procedura zpětného chodu, zpětná, cílem řízená inference
    - pro vybraný fakt hledáme předpoklady a pravidla, jejichž je důsledkem.

Základní problematikou procedur inference je problematika příkazu *vyber*. Čili problematika výběru pravidla, které budeme zpracovávat. Příkaz vyber se aplikuje na aktivní pravidla, množinu {P} a na otevřená fakta BF, přičemž otevřený fakt je takový, který ještě v procesu nebyl využitý, tj. pro fakt, který může způsobit aktivaci nějakého pravidla. Realizace tohoto příkazu závisí na strategii vložené do inferenčního mechanismu. Určuje funkční kvalitu IM a jeho realizace imituje kognitivní procesy experta.

Nejjednodušší principy operace vyber jsou založeny na hrubé síle. Nikterak nezohledňují podstatu řešeného problému a jen slepě prohledávají stavový prostor do *hloubky* nebo do *šířky*.

Snaha vylepšit tyto postupy pomocí lehce realizovatelných heuristik neměly valného účinku. Jednoduché heuristiky jsou například:

- Vyber poslední fakt vložený do BF
- Vyber fakt, který aktivuje nejmenší (největší) počet podmínek
- Vyber fakt v posloupnosti uspořádání v BF
- Vyber pravidlo, které má nejmenší (největší) počet podmínek
- Vyber pravidlo, na jehož interpretaci je v BF největší dostupný počet faktů
- Vyber pravidlo podle pořadí aktivace
- Vyber na vyhodnocení tu z požadovaných entit z množiny {P}, která je důsledkem největšího (nejmenšího) počtu pravidel
- Vyber tu entitu z množiny {P}, ke které vede nejkratší cesta ve stavovém prostoru

Uvedené heuristiky jsou velmi obecné a proto slabé. Opírají se pouze na charakteristiky symbolů a jejich uspořádání a nepostihují jejich obsah ani podstatu řešeného problému.

Problematika vyber se orientuje na využívání reprezentovaných poznatků. Získáváme tak heuristiky založené na:

- Specifičnosti faktu/pravidla
    - Čím větší specifičnost, tím větší možnost ohraničení stavového prostoru na jeho plausibilní části
- Senzitivitě faktu/pravidla
    - Čím větší senzitivita, tím rychlejší reakce systému na měnící se charakteristiky řešení problému
- Upřednostňování faktů odpovídajících nutné a postačující podmínce na odvození důsledku před těmi, které nejsou nutné či postačující
- Upřednostňování kategorických faktů před nejistými
- Nahrazování lokálních pohledů na řešení globálními
    - Lokálně nemusí být zřejmé souvislosti viditelné z globálního hlediska
- …

# 10 Další složky expertních systémů

V této přednášce se budeme věnovat dalším funkčním celků, modulům, znalostních systémů.

## 10.1 Komunikační modul (KM)

Komunikační modul zabezpečuje interakci mezi systémem a uživatelem. Jeho úkolem je zajistit jednoduchou, přirozenou, srozumitelnou a pohodlnou komunikaci. Mezi hlavní funkce komunikačního modulu patří:

- Inicializace a ukončení činnosti ES jako celku,
- inicializace činnosti modulů,
- realizace dialogu s uživatelem v průběhu inference:
    - předkládá uživateli dotazy, v případě, že je třeba se dotázat na nějaký fakt potřebný pro běh inference,
    - načítá odpovědi uživatele a kontroluje jejich správnost,
    - vypisuje chybová, či jiná hlášení, čímž usměrňuje činnost uživatele,
    - načítání dat zadaných z iniciativy uživatele,
    - modifikaci či rušení položek v bázi faktů,
    - přerušení, nebo ukončení činnosti inferenčního mechanismu,
    - obsloužení volitelných funkcí IM, změnu řídících parametrů, modifikaci priorit
- obsluhuje příkazy a požadavky uživatele.

Při inicializaci ES se specifikují služby, které od něj uživatel očekává, nejčastěji řešení problémů, nebo tvorba či modifikace báze znalostí.

Některé ES umožňují specifikovat jednak formu a rozsah interakce s uživatelem, a jednak způsob, rozsah a detaily inference v závislosti na možnostech systému.

Na začátku inference je třeba specifikovat inicializační fakta a určit cíl inference, případně požadované restrikce na výsledek. Je třeba brát v potaz, že ne každý uživatel má možnost získat jistá specifická fakta, jako například doktor v terénu nebude schopen diagnostickému ES dodat rentgen plic pacienta. Tato omezení je třeba specifikovat na začátku inference.

Komunikační modul by měl být schopen diferenciace přístupu k různým uživatelům. Při získávání faktů je zapotřebí klást odlišné dotazy uživateli zběhlému v používání expertních systémů, stejně tak by měl být schopen rozlišovat mezi uživatelem laikem a uživatelem odborníkem v dané oblasti.

V případech, kdy je interakce s uživatelem potlačena, jako je např. načítání dat z externích zdrojů, nebo případy, kdy byla při inicializaci zadána všechna potřebná data, se může komunikační modul omezit pouze na zasílání zpráv.

Komunikace s uživatelem a prostředí komunikačního modulu by mělo být uživatelsky přátelské a ergonomické.

Komunikační modul může být vyzbrojen povelovým jazykem, jako prostředkem pro komunikaci.

## 10.2 Vysvětlovací modul (VM)

Vysvětlovací modul zajišťuje vysvětlení rozhodování a usuzování ES. Na základě jeho předložení výsledku má uživatel možnost pochopit a racionálně odvozené výsledky přijmout nebo zamítnout. I přes nepochybný význam, není nezbytnou součástí ES. Existují aplikace, ve kterých je, stejně jako v případě komunikačního modulu, činnost vysvětlovacího modulu nežádoucí. Koncepce vysvětlovacího modulu může být, v závislosti na koncepci BZ a reprezentace poznatků v ní, velmi široká. Může obsahovat velmi jednoduché nebo velmi propracované vysvětlovací postupy.

Vysvětlovací modul běl by být schopen odpovědět na dva typy otázek.

- Proč:
  - požaduje daný fakt,
  - odvozuje daný fakt,
  - volí určitý způsob odvození,
  - vysvětluje co se děje nebo se předpokládá, že se bude dít v navazující inferenci.

- Jak:
  - dospěl k odvození určitého faktu,
  - dospěl k danému výsledku,
  - dospěl k danému způsobu odvození,
  - vysvětluje co, jak a z jakého důvodu se dělo v předchozí inferenci.

V souvislosti s hloubkou schopnosti objasňovat výsledky, může být VM schopen odpovídat i na další typy otázek:

- proč ne,
- co když,
- co je výhodnější,
- jaké jsou alternativy,
- v čem spočívá nejistota/riziko.

Schopnost kvalifikovaně odpovídat zvyšuje použitelnost a atraktivnost ES.

## 10.3 Generátor výsledků (GV)

Generátor výsledků předkládá výsledek inference ve vhodném tvaru. V průběhu inference může docházet k doplňování faktů, které jsou následně označeny za neplatné, mohou se procházet alternativní řešení, použít neúspěšné heuristiky atp. Mnohé postupy se posléze mohou ukázat jako zavádějící a špatné. Při předkládání výsledku inference uživateli není žádoucí takové postupy prezentovat. Zdůvodnění výsledku by se mělo zakládat na uvedení získaných či odvozených faktů, které k nim vedly. Výsledek by měl být tvořený určitou konfigurací faktů nebo entit ve vzájemných relacích. Generátor výsledků redukuje redundantní, zbytečné, zavádějící, nevhodné, špatné kroky inference a tím vzbuzuje představu přímočarého řešení. Pokud je to vhodné, zajišťuje generalizaci/specializaci faktů vedoucích k řešení. Jak jsme již uváděli, jsou případy, kdy nás nezajímá, jestli se jedná o kočku nebo delfína a stačí nám uvést pojem „savec“, v jiném případě zase požadujeme vědět, zda se jedná např. o tchoře, či fretku, samičku či samečka, což je v případě výběru domácího mazlíčka poměrně podstatná informace. Generátor výsledků zajišťuje odstranění nadbytečných, rušivých údajů ve výpise výsledku a integruje parciální výsledky a vybraná fakta do uspořádaného a srozumitelného celku.

## 10.4 Doplňkové složky ES

Některé znalostní systémy mohou být vybaveny i dalšími moduly.

### 10.4.1 Modul externích údajů (MEÚ)

Některá data potřebná pro běh inference mohou vytvářet externí zdroje, jako např. teplota ve vysoké peci, nebo v jádru reaktoru. Tato data umožňuje inferenčnímu mechanismu získat modul externích údajů. MEÚ je aktivován v případě, že IM potřebuje načíst data z okolí. V takovém případě MEÚ přebírá řízení systému a zabezpečuje prohledávání externích dat. Při nenalezení potřebného údaje může být prostřednictvím komunikačního modulu vyzván uživatel, aby údaj doplnil.

### 10.4.2 Modul externích programů (MEP)

Obdobně, jako MEÚ zajišťuje komunikaci s vnějšími zdroji, modul externích programů zajišťuje interakci systému s externími programy.

Modul externích programů tvoří interface mezi systémem a externími programy, může vykonávat reproduktivní algoritmy potřebné pro řešení problému, volat externí programy a interpretovat jejich data.

### 10.4.3 Moduly pro práci s BZ

Jak jsme již uvedli v předešlých přednáškách, tvorbu báze znalostí lze od tvorby znalostního systému oddělit. V případě prázdných expertních systémů je nezbytně nutné, pro jejich používání v některé oblasti, bázi vytvořit. V průběhu testování znalostních systémů můžeme požadovat provádění změn v BZ. BZ znalostí je třeba kontrolovat atd. K tomu je vhodné vlastnit nástroje, které nám práci usnadní. Procházet stovky či tisíce pravidel zapsaných v nějakém programovacím jazyce či jiném formálním nástroji a hledat chyby, může být nadlidský úkol.

### 10.4.4 Modul definování BZ (MD)

Modul definování BZ slouží znalostnímu inženýrovi, programátorovi či přímo expertovi k naplňování BZ, v souladu s použitým formalismem znalostí. MD by měl kontrolovat syntaktickou i sémantickou správnost zadávaných struktur a vést tvůrce báze k co nejúplnější reprezentaci tím, že mu automaticky a smysluplně předkládá odpovídající možnosti ve smyslu formální reprezentace poznatků.

### 10.4.5 Modul modifikace BZ (MM)

Ať už v průběhu tvorby báze znalostí či její revize v průběhu testování nebo provozu systému mohou vznikat požadavky na její modifikaci. Modul modifikace BZ má, jak svým určením, tak funkcí, velmi blízko k modulu definování BZ. Poskytuje interaktivní prostředky na modifikaci či rušení existujících položek báze znalostí.

### 10.4.6 Modul kontroly BZ (MK)

Jak MD, tak MM plní částečně i kontrolní funkce, jejich schopnost kontroly je omezená a zaměřuje se především na syntaktickou a sémantickou kontrolu. Rozsáhlé báze znalostí však vyžadují mnohem důslednější a hlubší kontrolu, kterou zajišťuje modul kontroly BZ. MK by měl být schopen, krom kontroly celé báze znalostí, na základě parametrizace kontrolovat pouze určité definované části. Tato částečná kontrola sice proběhne rychle, ale nese sebou riziko neodhalení některých chyb.

Modul kontroly by měl být schopen, krom klasické syntaktické a sémantické kontroly, odhalit chyby typu:

- nedefinované objekty – v BZ se odvoláváme na objekt, který v ní není definovaný,
- nadbytečné objekty – v BZ se vyskytují objekty, které nejsou ve vztahu s jinými objekty, čili nikdy nemohou ovlivnit průběh inference,
- kontradikce – podmínky, které vzhledem na reprezentovanou realitu nemohou být nikdy splněné,
- tautologie – za všech okolností splnitelná – pravdivá podmínka,
- nedostupné objekty – v BZ je definován objekt, který nemůže být nikdy v průběhu inference dosažitelný – je dosažitelný pouze na základě nesplnitelné podmínky,

- typová neshoda objektů – v BZ se vyskytuje relace mezi objekty, které si typově neodpovídají (např. boolean a integer),
- nesprávné hierarchické zařazení – objekt je v určité generalizačně-specializační hierarchii jak generalizací, tak specializací jiného objektu,
- nepřípustné cyklické volání – přímé, nebo nepřímé volání entity sebe samou. O entitě je tak možno rozhodnout pouze tehdy, pokud by byl k dispozici fakt, který by byl známý pouze tehdy, pokud by už bylo o entitě rozhodnuto a v dané reprezentaci neexistuje jiná možnost ověření takové entity,
- a další, závislé na reprezentaci, či požadavkům na kontrolu BZ.

### 10.4.7 Modul prohlížení BZ (MP)

Někdy samotné odstranění chyb pomocí MK nemusí být dostatečné pro správný chod systému. V takovém případě je nutné důkladně projít bázi znalostí a dohledat případné chyby. Pro BZ tvořené jednoduchými strukturami není prohlížení BZ problém. Pokud však máme BZ velmi rozsáhlou, tvořenou složitými strukturami, je vhodné vlastnit nástroj, který nám pohled do BZ učiní co nejvíce přehledným. Získáme tak přehled o reprezentovaných znalostech, jejich detailech, členění i začlenění. Nástroj, který nám to umožní je modul prohlížení BZ. Prohlížení obsahu BZ má význam především při odhalování příčin neadekvátního chování systému, případně při přípravě doplňků a modifikací, zvláště pokud je vhodné mít ucelený přehled na obsah a strukturu BZ.

Mezi nejčastější funkce MP patří:

- výpis přehledu reprezentovaných entit
- celkový, nebo částečný obsahu BZ
- prohledávání všech, nebo specifických vzájemných volání reprezentovaných entit
- prohledání všech, nebo konkrétních odvozovacích posloupností
- výpis vnitřní formy reprezentace BZ

# 11 Vývoj znalostního inženýrství

S vývojem znalostních systémů, hromaděním zkušeností s vývojem znalostních systémů a zkušeností spolupráce s experty vzniká v IT nový obor – znalostní inženýrství, které se zabývá problematikou efektivní tvorby ZS.

První období rozvoje spadá do druhé třetiny 70. let do poloviny 80. let 20. století. V tomto období dochází především k vytváření programátorských technik tvorby znalostních systémů. Ustaluje se architektonické členění na bázi znalostí, bázi faktů, inferenční mechanismus, komunikační modul atd., což umožňuje systematizovat postupy tvorby a dekompozici na podproblémy, které je třeba vyřešit. V souvislosti s tvorbou báze znalostí dochází do jisté míry ke kodifikaci reprezentace poznatků, a to především pomocí produkčních pravidel a rámců. Dochází taktéž ke standardizaci cílem a fakty řízeného odvození, jako postupů inference. Při získávání poznatků od expertů se v prvním období uplatňuje bezprostřední převod znalostí do reprezentačních schémat. Tento postup je využíván i dnes, a to v případě projektů menšího rozsahu, které obsahují pouze několik stovek poznatků, a nejedná se o investiční charakter projektu. Tento postup nazýváme *prototypové řešení*, protože bezprostředně po zakódování poznatků je de facto vytvořen prototyp.

V druhém období, které bylo podnícené studií Allena Newella (1982), kdy se poprvé vyskytuje oddělení aktivit spojených s *úrovní poznatků*, kam patří identifikace, získávání a reprezentace poznatků, od *úrovně výpočtů*, do které spadají implementační problémy. Dochází ke sjednocení terminologie. Utváří se pojmové schéma problematiky tvorby ZS, které je nezávislé na implementačních technikách. Strohý převod poznatků nahrazuje modelování znalostí experta. Za výsledek práce znalostního inženýra se přestává považovat funkční prototyp, ale cílem je vytvořit komplexnější výpočtově realizovaný model profesionálních aktivit experta. Tyto modely v sobě zahrnují zejména soustavu poznatků, včetně vyjádření neurčitosti, jejich hierarchické či kauzální vztahy a charakterizaci postupů. Které expert při řešení problému používá. To má ovšem za následek rozestupy mezi konceptuálními modely a implementací.

# 12 Životní cyklus ZS

Tvorba znalostního systému, stejně jako tvorba jakéhokoliv jiného informačního systému by se měla držet jasného explicitního rámce. V dnešní době existuje řada metod a metodik tvorby informačních systémů. Drtivá většina má společné tři hlavní fáze, analýzu, vývoj a provoz a údržbu. My uvedeme schéma KLIC – Knowledge-based System Life Cycle (viz. Giovany Guida, Carlo Tasso: Design and Development of Knowledge-Based Systems, John Wiley & Sons, New York, 1994). KLIC se skládá ze tří makrofází a šesti fází. Životní cyklus projektu znalostního systému představuje úplný popis jednotlivých fází tvorby znalostního systému – od úvodní analýzy až po reálné používání. Musí mít jasnou strukturu postavenou na principu hierarchické dekompozice problému. Chápeme jej ve dvou rovinách:

- vyšší úroveň abstrakce – fáze
- nižší úroveň abstrakce – úlohy.

Pro větší transparentnost a explicitu se fáze seskupují do makrofází a úlohy do kroků.

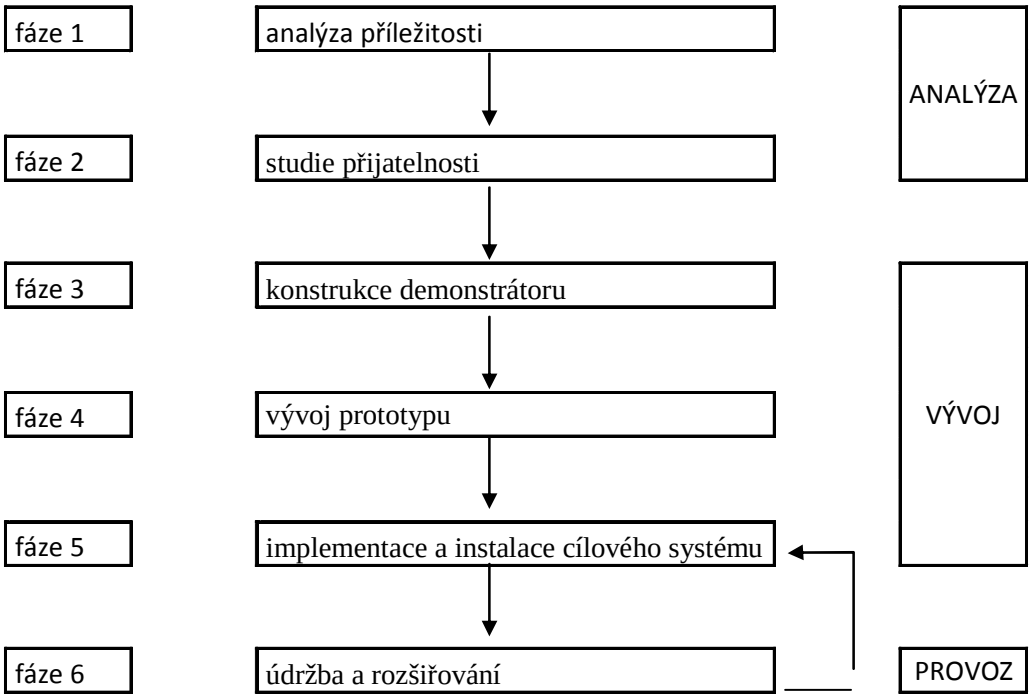



**Schéma životního cyklu KLIC**

## 12.1 Fáze životního cyklu ZS

**Fáze 1 - analýza příležitostí**

Před začátkem každého projektu je vhodné nejdříve udělat průzkum, jehož cílem je najít a ohodnotit budoucí možnosti potencionálního znalostního systému. Průzkum začíná identifikací aplikačních oblastí, ve kterých které je vhodné použití znalostních technologií. Identifikované příležitosti se hodnotí podle:

- strategické hodnoty
- taktické důležitosti
- očekávaných přínosů
- technické složitosti
- vhodnosti
- rizika

Tato fáze se netýká přímo vývoje znalostního systému, vytváří však podklady pro aplikaci konkrétních projektů. Výsledkem analýzy by měl být písemně zpracovaný materiál, ve kterém jsou specifikované aplikační oblasti, které je v daném podniku (anebo jiné instituci) vhodné řešit na bázi znalostních technologií. Fázi analýzy vykonává zpravidla nezávislý konzultant v úzké spolupráci s manažerem a jinými pracovníky objednávající organizace. Ve středních a malých podnicích se doba vykonání této fáze odhaduje na 3 až 9 měsíců, ve velkých organizacích je to 10 až 15 měsíců. Tato fáze sice není povinné, ale její provedení se velmi doporučuje. Odhalená přílišná rizika, či nevýhodný poměr očekávaného přínosu oproti nákladům mohou ušetřit nemalé náklady na vývoj systému.

**Fáze 2 – studie přijatelnosti**

Základním cílem této fáze je:

- analýza aplikační oblasti, identifikované a vybrané v předchozí fázi
- analýza požadavků a komplexních cílů projektu
- stanovení základních funkčních, operačních a technických specifikací znalostního systému a kritérií přijatelnosti
- vývoj hrubého technického návrhu, organizačního návrhu a plánu projektu.

Koncept přijatelnosti zahrnuje pět aspektů:

- technickou vhodnost

- organizační důsledky
- ekonomickou vhodnost
- prektickou realizovatelnost
- příležitost a riziko

Výsledkem by opět měl být písemně vypracovaný materiál, technický dokumentem pro management. Popisuje vykonané činnosti, dosáhnuté výsledky a doporučuje alternativy pro rozhodovací proces. Fázi vykonává nezávislý externí konzultant ve spolupráci s managementem a pracovníky organizace, jen ve výjimečných případech interní pracovník podniku. Na rozdíl od předcházející fáze je vykonání této fáze povinné a její trvání se odhaduje na 1 až 2 měsíce v malých podnicích, na 3 až 6 měsíců ve středních podnicích a na 7 až 9 měsíců ve velkých organizacích.

**Fáze 3 – konstrukce demonstrátoru**

Základním cílem této fáze je vývoj a demonstrace první, velmi zjednodušené verze znalostního systému. V této fázi si mohou zúčastnění pracovníci vytvořit jasnou představu o složitosti celého problému, co může vést k revizi technických rozhodnutí, plánu projektu, definice a specifikace znalostního systému z předcházející fáze. Velmi podstatná je zpětná vazba od budoucího uživatele, který má možnost si otestovat práci se znalostním systémem, zhodnotit ergonomii a uživatelskou příjemnost prostředí systému.

Tato fáze bývá kritickým bodem úspěšnosti projektu. V závislosti na tom, jaký dojem udělá první verze systému na management, pokračují v podpoře dalšího vývoje. Už v této fázi se projeví i zájem (resp. nezájem) o spolupráci ze strany expertů a uživatelů. Důležité je dostatečně objasnit, zejména managementu, že demonstrátor není finálním produktem a rozhodně nesplňuje veškerá zadaná kritéria.

Výsledkem fáze je fungující verze znalostního systému tzv. demonstrátor, neboli demonstrační verze, která představuje budoucí systém v intensích malé části uvažovaného problému a písemný materiál, který popisuje vykonané aktivity a dosáhnuté výsledky.

Konstrukci demonstrátoru většinou vykonává externí vývojový tým, ve spolupráci s interními zaměstnanci uživatelské organizace. Doba trvání se pohybuje od 2 do 6 měsíců v závislosti na složitosti aplikace a cíle demonstrátoru. Tato fáze životního cyklu není povinná.

**Fáze 4 – vývoj prototypu**

**C**ílem této fáze je najít nejvhodnější technické řešení aplikace a její implementace do fungujícího systému organizace. Výsledkem je prototyp znalostního systému, který plní všechny funkční charakteristiky specifikované v předcházejících fázích a integrovaný soubor softwarových nástrojů, vývojový podpůrný systém, který ulehčuje tvorbu báze znalostí z prototypu a písemný materiál, který popisuje vykonané aktivity a dosáhnuté výsledky.

Přestože prototyp splňuje funkční požadavky, není ještě konečným výstupem vývojového procesu. Běží jen ve vývojovém prostředí a případné propojení s externím světem musí být simulované, jeho testování proběhlo jen na datovém vzorku, který vytvořil návrhář systému ve spolupráci s expertem a uživateli.

Prototyp většinou vytváří externí vývojový tým ve spolupráci s pracovníky uživatelské organizace v trvání 6 až 24 měsíců v závislosti na složitosti aplikace. Vývoj prototypu je povinná fáze životního cyklu znalostního systému.

Prototyp je ve všeobecnosti systém úplně odlišný od demonstrátoru. Cíle, principy návrhu a vývojové nástroje použité pro tyto dva systémy mohou být natolik odlišné, že inkrementální vývoj prototypu z demonstrátora nemusí být vhodný ani pohodlný.

**Fáze 5 – implementace a instalace cílového systému**

Cílem této fáze je vývoj kompletního znalostního systému, jehož chování je totožné s prototypem, je však nainstalovaný v reálném prostředí. Testování probíhá na reálných datech a jeho činnost musí být optimalizována.

Výsledkem této fáze cílový systém, který je konečným produktem vývojového procesu znalostního systému, kompletní soubor manuálů pro uživatele. Písemná zpráva z této fáze životního cyklu popisuje vykonané aktivity a popisuje dosáhnuté výsledky.

Implementace cílového systému může vyžadovat rozdílné přístupy v závislosti na požadavcích a podmínkách stanovených prostředím – od automatického generování konečné verze přes specifické podpůrné nástroje až po inkrementální vývoj systému z prototypu, případně – v nejhorším případě – kompletní reimplementaci systému.

Fázi implementace a instalace vykonává buď interní projektový tým organizace, pro kterou byl systém vyvíjený anebo externí dodavatel. Doba trvání implementace a instalace je obvykle 4 až 24 měsíců, v závislosti na přístupu k implementaci.

**Fáze 6 – údržba a rozšiřování**

Fáze údržby a rozšiřování se začíná po uvedení znalostního systému do reálného používání a trvá po dobu celého jeho „života“. Na kvalitě poskytovaných služeb v této fázi závisí délka a efektivnost používání systému a využití všech potencionálních přínosů projektu.

Mezi základní cíle této fáze patří:

- monitorování provozu systému a zpětné vazby s uživatelem (jeho požadavků a připomínek)
- oprava chyb
- případná aktualizace systému, aby vyhovoval změnám hardware a software, případně změnám v požadavcích uživatelů – tzv. evoluční údržba

Reálným výsledkem 6. fáze jsou nové verze báze znalostí, upravené manuály, nové verze cílového systému a systému na podporu a údržby a materiál, který předběžně zachytává připomínky, poznámky a požadavky uživatelů a záznamy o všech aktivitách, které byly vykonané ve fázi údržby a rozšíření.

Údržbu vykonává zpravidla speciální tým pracovníků, kteří jsou touto úlohou pověřeni. Rozšíření vykonává větší projektová skupina (ať už externí nebo interní). Obě aktivity by měly být vykonané bezprostředně po identifikaci jejich potřeby. Každý zásah do báze znalostí (s cílem údržby) netrvá déle než 3 až 10 dní, rozšíření je časově podstatně náročnější – odhadovaný čas se pohybuje v rozsahu od 1 do 8 měsíců. Jde o povinnou fázi životního cyklu, kterou v žádném případně není možno vynechat.

# 13 Základy teorie fuzzy logiky a fuzzy množin

Fuzzy

- marked by or giving a suggestion of fuzz <a *fuzzy* covering of felt>
- lacking in clarity or definition <moving the camera causes *fuzzy* photos>
- being, relating to, or invoking pleasant and usually sentimental emotions <warm and *fuzzy* feelings>
- nejasný, mlhavý, chmýřovitý, neurčitý, chomáčkovitý, chlupatý, zmatený, konfuzní, konfúzní, kučeravý, kudrnatý, mající dlouhé chlupy, nakadeřený, nalíznutý, nehomogenní, neostrý, rozmazaný, roztřepaný

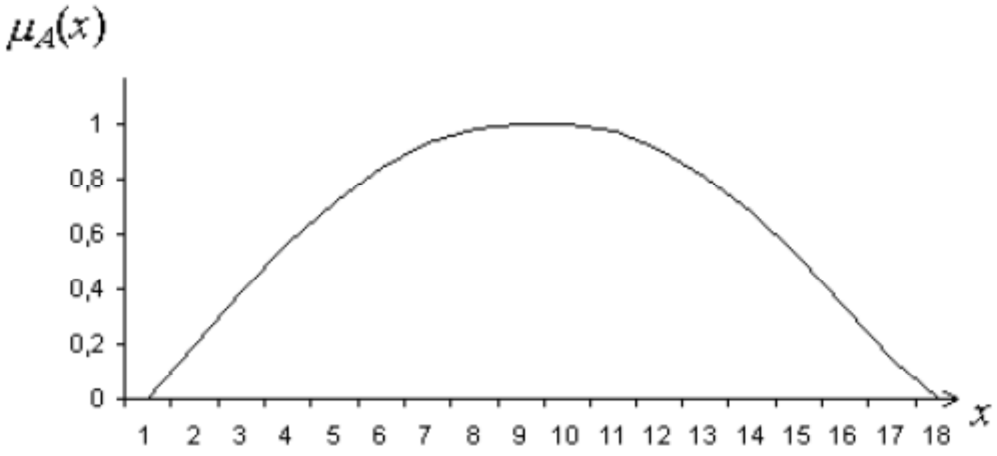
Jak je vidět, slovo *fuzzy* má mnoho významů. Ve spojení s matematikou, logikou, informatikou atp. je překládáno jako nejistý, neurčitý. S neurčitostí se setkáváme na každém kroku. Je základním znakem komunikace v přirozeném jazyce. Každý z nás rozumí pojmu malý člověk, mladý člověk, rychlé auto. Ale jaký člověk je malý? Když se pokusíme tento pojem zachytit pomocí klasické teorie množin, tak musíme přesně stanovit, kdo do množiny patří. Řekneme-li že malý člověk je ten, který měří do 160 centimetrů, pak člověk měřící 160,1 cm už do množiny patřit nebude, přestože je rozdíl jednoho milimetru ve výšce okem nerozeznatelný. Z jiného hlediska, člověk měřící 180 cm by mohl být v Evropě považován ještě za, řekněme, středně vysokého. V Japonsku už určitě dostane takový člověk přídomek *vysoký*. Neurčitost, se kterou v lidském jazyce pracujeme, odráží mimo jiné i regionální zvyklosti. Stejný pojem může mít v různých regionech různý význam.

Snaha matematicky zachytit a popsat neurčitost vyústila v zavádění fuzzy teorií. V roce 1965 Lotfi A. Zadeh ve svém článku *Fuzzy sets* publikovaném v *Information and Control*, zavádí fuzzy množiny.

Fuzzy množiny byly zavedeny pro reprezentaci a manipulaci s daty a informacemi, které obsahují statisticky nepopsatelnou neurčitost. Cílem této teorie bylo matematicky reprezentovat *vágní* data a stanovit formální nástroje pro práci s nepřesností. Základním rozdílem mezi klasickou teorií množin a teorií fuzzy množin je příslušnost prvku k množině. Zatímco v klasické teorii prvek x množině A buďto náleží, $x \in A$, nebo nenáleží, $x \notin A$, v teorii fuzzy množin určuje příslušnost prvku x množině A funkce příslušnosti $\mu_A(x)$. Místo $\mu_A(x)$ lze

psát pouze A(x). Prvek tedy množině náleží s jistou mírou příslušnosti. Funkce příslušnosti přiřazuje každému prvku číslo z intervalu [0,1]. Fuzzy množina je tedy zobecnění klasického pojmu množina. Formálně:

- Nechť U je neprázdná množina. Fuzzy množina A v U je dána funkcí příslušnosti $\mu_A$:U→[0, 1], kde $\mu_A(x)$ je míra příslušnosti prvku x k fuzzy množině A, pro všechna x∈U. Fuzzy množina A je podmnožinou universa U a je definována jako množina uspořádaných dvojic A={x, $\mu_A(x) \mid x \in U$}.
- Prázdná fuzzy množina ∅ je definována funkcí příslušnosti $\mu(x) = 0$ všechna x ∈ U.
- Nosič (support) fuzzy množiny A je klasická množina $supp(A) = \{x \mid \mu_A(x) > 0\}$
- Jádro (kernel) fuzzy množiny A je klasická množina $ker(A) = \{x \mid \mu_A(x) = 1\}$
- Výška (height) fuzzy množiny A je definována takto $hgt(A) = \sup\{\mu_A(x) \mid x \in U\}$




**Příklad funkce příslušnosti**

Neurčité údaje jsou často reprezentovány jako fuzzy čísla. Fuzzy číslo A je normální konvexní fuzzy množina na universu reálných čísel, která je určena čtveřicí bodů ($a_1$, $a_2$, $a_3$, $a_4$) a po částech spojitou funkcí příslušnosti s následujícími vlastnostmi:

- $a_1 \leq a_2 \leq a_3 \leq a_4$
- $\mu(x) = 0$, pro $x \leq a_1$, $x \geq a_4$
- $\mu(x) = 1$, pro $a_2 \leq x \leq a_3$
- $\mu(x)$ je rostoucí v intervalu $[a_1, a_2]$, klesající v $[a_3, a_4]$

Mějme fuzzy množiny A=(U, $\mu_A$), B=(U, $\mu_B$), základní operace s fuzzy množinami definujeme takto:

- Doplněk fuzzy množiny $\overline{A}$: $\overline{A} = (U,\ \mu_{\overline{A}}),\ \mu_{\overline{A}}(x) = 1 - \mu_A(x)$
- Sjednocení fuzzy množin A a B: $A \cup B = (U, \mu_{A\cup B}), \mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$
- Průnik fuzzy množin A a B: $A \cap B = (U, \mu_{A\cap B}), \mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$

Vztahy rovnost a inkluze jsou definovány takto:

- A = B, právě když $\mu_A(x) = \mu_B(x)$ pro všechna $x \in U$
- A ⊆ B, právě když $\mu_A(x) = \mu_B(x)$ pro všechna $x \in U$
- A ⊂ B, právě když $\mu_A(x) = \mu_B(x)$ pro všechna $x \in U$

Z teorie fuzzy množin vychází fuzzy logika. Obdobně jako v případě fuzzy množin, je každému výroku $p$ přiřazena pravdivostní hodnota $val(p) \in [0, 1]$. Jednotlivé operace definujeme následovně:

- Fuzzy negace – unární operace ¬: $[0, 1] \to [0, 1]$, pro kterou platí:
  - $\neg\neg p \equiv p$
  - $val(p) \leq val(q) \Rightarrow val(\neg p) \geq val(\neg q)$
  - $val(\neg p) = 1 - val(p)$
- Fuzzy konjunkce – binární operace ∧: $[0, 1]^2 \to [0, 1]$, pro kterou platí:
  - $p \wedge q \equiv q \wedge p$ – komutativnost
  - $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$ – asociativnost
  - $p \wedge 1 \equiv p$ – identita
  - $val(q) \leq val(r) \Rightarrow val(p \wedge q) \leq val(p \wedge r)$
  - $val(p \wedge q) = \min\{val(p), val(q)\}$
- Fuzzy disjunkce – binární operace ∨: $[0, 1]^2 \to [0, 1]$, pro kterou platí:
  - $p \vee q \equiv q \vee p$ – komutativnost
  - $p \vee (q \vee r) \equiv (p \vee q) \vee r$ – asociativnost
  - $p \vee 0 \equiv p$ – identita
  - $val(q) \leq val(r) \Rightarrow val(p \vee q) \leq val(p \vee r)$
  - $val(p \vee q) = \max\{val(p), val(q)\}$

Posledním pojmem, který si v souvislosti s úvodem do fuzzy teorie uvedeme je *jazyková proměnná*. Je to proměnná, jejíž hodnotou mohou být slova nebo výrazy přirozeného jazyka. Jazyková proměnná je obecně definována jako uspořádaná pětice:

- $\chi = (X, T, U, G, M)$, kde
  - X je jméno proměnné (věk)
  - T je množina termů – slovních hodnot, které může proměnná nabývat (mladý, starý, spíše mladý, spíše starý…)
  - U je univerzum (neprázdná klasická množina)
  - G je množina syntaktických pravidel pro generování hodnot z T
  - M je množina sémantických pravidel interpretujících hodnoty z T

## 13.1 Fuzzy pravidla typu IF-THEN

Fuzzy pravidla typu IF-THEN definujeme: $R :=$ IF $X$ is $A$ THEN $Y$ is $B$, kde A, B jsou fuzzy množiny, nebo fuzzy čísla. Fuzzy pravidla hrají roli v každodenním lidském úsudku. V úvodu přednášky jsme zmínili příklad s malým člověkem. Příkladem z každodenního života, kdy užíváme fuzzy pravidla, aniž bychom tento pojem znali, může být např. přecházení cesty mimo přechod pro chodce, v konečném důsledku i na něm. Než přejdeme silnici, měli bychom se důkladně rozhlédnout. Jestliže jede nějaké auto, zvažujeme, nebo bychom měli zvažovat (mnoho chodců tak nečiní a vystupují v roli sebevraha…), hned několik parametrů. Vzdálenost, rychlost, povrch a stav vozovky atd. Na základě toho, jestli je auto *dostatečně daleko*, *nejede moc rychle*, silnice je *celkem suchá*, a tudíž brzdná dráha je krátká, se rozhodneme, jestli silnici přejdeme nebo počkáme, až přejede. Veškerá fuzzy pravidla, která jsme použili v tomto příkladu *jazykový popis* daného problému. Čili jazykový popis je množina všech fuzzy pravidel popisující daný problém. Jak je z příkladu patrné, jazyková část fuzzy pravidla je modelována pomocí fuzzy množin. Interpretace a mechanismus pro odvození závěru z fuzzy pravidel, nebo jazykového popisu, je založen na fuzzy logice.

Fuzzy pravidla můžeme využít, krom rozhodování, zdali je bezpečné přejít cestu i na fuzzy aproximaci. Pomocí fuzzy pravidel charakterizujeme funkční závislosti v nepřesně specifikovaných oblastech. Například aproximace funkce, jejíž funkční hodnoty neznáme přesně. Průběh takové funkce můžeme popsat fuzzy pravidly, kde předpokladovou část

pravidla tvoří fuzzy množiny definičního oboru a důsledkovou část fuzzy množiny oboru hodnot aproximované funkce. Hovoříme o *funkcionální interpretaci*.

Zvláštní postavení mezi pravidly mají pravidla typu Takagi-Sugeno. Jedná se o speciální typ pravidel IF-THEN. Doposud jsme uvažovali pravidla, ve kterých je charakterizován vztah mezi nepřesně zadanými hodnotami závisle a nezávisle proměnných. Pravidla typu Takagi-Sugeno řadíme mezi podmíněná pravidla. Sukcedent pravidla je zadán lineárním vztahem na antecedentu. Pravidla Takagi-sugeno definujeme: $R \coloneqq \text{IF } X \text{ is } A \text{ THEN } Y = a + bX$ .

# 14 Fuzzy expertní systémy

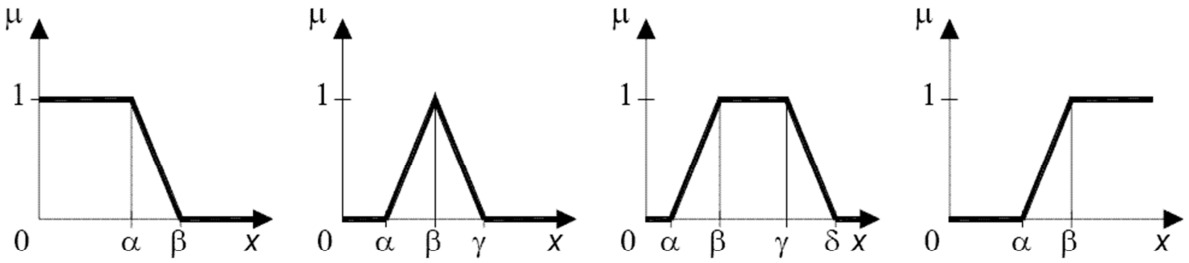
Jak jsme uvedli na začátku přednášky, neurčitost, vágnost je součástí každodenního života. Navíc je to charakteristický rys mnohých systémů. Data získaná z těchto systémů potřebná pro řešení problému jsou nekonzistentní, nepřesná, mnohá jsou nedostupná, nespolehlivá atd. Navíc mohou být i znalosti potřebné pro řešení takovýchto problémů vágně definovány nebo mohou platit pouze v některých (vágně určených) případech. Velmi názorným příkladem vágně definovaných znalostí je schopnost onkologa odhalit na rentgenu nádor. Ve škále šedi nalezne místo, které je méně, nebo jinak šedé. Vágní pojmy, které se navíc vztahují ke konkrétnímu snímku. Jinak exponovaný snímek, byť jen nepatrně, může mít škálovatelnost odlišnou. V úvodu kapitoly jsme definovali rozdíl mezi klasickou a fuzzy množinou. Při použití klasického množinového přístupu bychom byli nuceni buďto přesně definovat hranice toho, co ještě nádor není a co už nádor je, nebo použít pravděpodobnostních či statistických metod, jako je např. Bayesovský přístup, kde je neurčitost závěru H v závislosti na předpokladu E být kvantifikována pomocí podmíněné pravděpodobnosti P(H | E). Další možností je využití fuzzy přístupu. V příkladu s karcinomem se nám dokonce přímo vnucuje použití fuzzy pravidel typu Takagi-Sugeno.

Fuzzy přístup zavádí nejistotu do faktů a pravidel. Pravidla definujeme pomocí fuzzy množin a k práci s nimi využíváme prostředků fuzzy logiky a fuzzy množin. Inference probíhá odvozováním pomocí fuzzy pravidel.

Řešení problému pomocí fuzzy expertního systémů probíhá na základě čtyř základních operací:

- Fuzzifikace
- Inference
- Agregace
- Defuzifikace

Fuzzifikace je transformace konkrétních vstupních hodnot do normalizovaného tvaru a přiřazení stupně příslušnosti do jedné nebo více fuzzy množin. Je nezbytně nutné pokrýt zvolenými nosiči jednotlivých fuzzy množin celé normalizované univerzum, aby nedošlo ke ztrátě dat. Každý vstup tak bude přiřazen do alespoň jedné fuzzy množiny.




L – funkce, Λ - funkce, Π - funkce, Γ - funkce.

V průběhu inference se k zadaným fuzzifikovaným faktům se hledají předpoklady fuzzy pravidel, jejichž antecedent těmto faktům nejlépe vyhovuje a vyvozují se důsledky v podobě fuzzy množin. Výsledkem fuzzy inference je teda sada fuzzy množin. Tento výsledek je třeba převést do nějaké formy, která má vypovídací hodnotu. To zajistí poslední dvě operace.

Výsledkem odvozování je několik fuzzy množin. Tyto množiny agregujeme jedné fuzzy množiny. Agregace je tedy složení fuzzy množin – sukcedentů použitých pravidel, přičemž výsledkem je jedna fuzzy množina.

Z této jedné množiny pomocí defuzzifikace získáme reálné číslo. Defuzzifikace je transformace agregované fuzzy množiny na reálné číslo, přičemž se nejčastěji používají metody:

- COB (Center of Balance) – defuzzifikace pomocí těžiště plochy pod funkcí příslušnosti výsledné fuzzy množiny
- MOM (Middle of the Maximum) – defuzzifikace pomocí střední hodnoty lokálních maxim funkce příslušnosti výsledné fuzzy množiny

# 15 Kvalita expertních systémů

Stanovení kvality SW je problematické. Obecně lze říct, že kvalitu software stanovuje zákazník na základě naplnění jeho požadavků, kvalitě dokumentace a poskytnutém servisu. Pro stanovení kvality nicméně máme k dispozici dva nástroje:

- Ověřování – proces, ve kterém se určuje, zda produkt určité fáze vývoje splňuje požadavky stanovené v předchozí fázi.
- Hodnocení – proces, ve kterém se zjišťuje stupeň splnění specifikovaných požadavků na software na konci vývojového procesu.

Zákazník, tedy budoucí uživatel hodnotí systém z pohledu otázek:

- Splňuje předkládaný systém požadavky uživatele?
- Řeší skutečné problémy, které řešit má“?

Zákazníka teda zajímá použitelnost, kompetence, chování a spolehlivost systému.

Termín *tvůrce* znalostního systému pokrývá více druhů specialistů s různými úlohami (vývojář, znalostní inženýr a expert), kteří při tvorbě systému vzájemně úzce spolupracují. Tvůrce systému se zaměřuje v procesu ověřování na otázky typu:

- Funguje vytvořený softwarový systém korektně?
- Řeší problém zákazníka tak, jak to vyplynulo ze společných diskuzí?

Ověřování se týká inferenčního mechanismu, uživatelského rozhraní, poznatků a metapoznatků (poznatky o poznatcích). Tvůrce sleduje syntaktické vlastnosti jako úplnost, robustnost, korektnost a chování systému jako softwaru i jako experta.

Mezi hodnocením a ověřováním znalostního systému a hodnocením a ověřováním báze poznatků je zásadní rozdíl. Při ověřování a hodnocení systému nás zajímá software jako celek, v případě báze znalostí, soubor poznatků, který je však důležitou součástí softwaru. Ověřování znalostního systému znamená ověřování inferenčního mechanismu, báze znalostí a řídících toků, ověřování báze znalostí, např. pro pravidlový systém, v podstatě znamená potvrzení správnosti každého pravidla.

Základní technikou procesu hodnocení a ověřování informačních systémů obecně, je *testování*. Testování probíhá vesměs na množině testovacích dat, které obsahuje historické i aktuálních data. Základní otázkou ovšem je reprezentativnost takovýchto testovacích dat a z toho plyne otázka, je tedy IS dostatečně otestovaný? Problematika reprezentativnosti testovacích dat je zřejmá. Pro potřeby testování jsme schopni připravit data na základě již řešených problémů. Jsme schopni vytvořit extrémní či zřídkavé případy, které slouží k odhalení slabých míst systému. Nicméně těžko můžeme s jistotou říct, že i takto nasimulované extrémy jsou tím největším zatížením, kterým může systém v ostrém provozu projít.

Kvalita otestování systému nezávisí pouze na kvalitě testovacích dat. Spolu s kvalitními testovacími daty musíme mít i zkušené *testery*, kteří systém testují. Je třeba si uvědomit, že vývojář zná omezení systému, na jehož vývoji se podílel, zná procesy, které v něm probíhají a ví, co mu může systém nabídnout. Pozice uživatele je ovšem přesně opačná. Uživatel zkoumá a hledá hranice. Mnohdy ovšem odhalí procesní chyby, nebo zcela nelogickou posloupností akcí, jako např. několikeré stisknutí backspace ve webovém rozhraní aplikace a následné opakované odeslání již jednou odeslaných dat na serverovou část, způsobí chybové chování systému. Zkušení testeři znají mnoho uživatelských nástrojů sloužících k destrukci vývojářova snažení, nicméně, nutno říct, že BFU (Běžný Franta Uživatel, anglickou verzi tohoto akronymu, nechť si čtenář vygooglí sám…) užívající aplikaci mají velmi často navrch i před nejzkušenějšími testery.

Testování se vesměs provádí v simulovaném prostředí na *testovací instanci* aplikace, které je držena mimo ostrý provoz a ostrá data. Zanedbání testování systému může mít nedozírné následky. Od pouhé nefunkčnosti aplikace v ostrém provozu, přes celkový kolaps informačních systémů podniku až po nevratnou ztrátu dat.

## 15.1 Hodnocení

Vzhledem k celkové složitosti problematiky hodnocení znalostních systémů bylo vyvinuto množství metod, které proces sytematizují a tak značně pomáhají tomu, kdo hodnocení vykonává:

1) Testování na reálných datech je způsob testování s daty, s jakými bude systém pravděpodobně pracovat, když bude uvedený do provozu. Hodnotitel tak může přímo sledovat chování systému v (simulovaných) reálných situacích.
2) Testování na speciálních datech je způsob, jak odhalit slabá místa v systému, zpravidla na extrémních anebo nečekaných, zřídka se objevujících datech. Nalezení speciálního testovacího souboru je výsledkem značného úsilí, které nemusí být rezultativní. Takto vznikla myšlenka využití inteligentních systémů na tvorbu speciálních testovacích dat.
3) Přímé zkoumání báze znalostí. Využívá ho expert v případě, že poznatky jsou v bázi znalostí uložené v čitelné formě, nebo vlastníme nástroje na prohlížení báze znalostí. Když analogickou aktivitu vyvine znalostní inženýr, není tato forma hodnocení dostačující kvůli nedostatečné znalosti problematiky.
4) Paralelní použití – ve stejném čase řeší stejný problém systém i expert, případně uživatel, který není expertem v dané oblasti (právě jeho řešení může být podnětné a inspirativní)
5) Statistické metody – procento správně řešených problémů atp.

Hodnocení znalostních systémů vykonává buď expert, jehož poznatky jsou v systému uložené, jiný, nezávislý expert, znalostní inženýr, koncový uživatel anebo nezávislý hodnotitel.

Hodnocení experta – tvůrce BZ:

- výhoda – zná systém a umí se v něm orientovat, už jednou byl nucen systematizovat poznatky uložené v systému,
- nevýhoda – může lehce přehlédnout případnou nekonzistentnost či neúplnost.

Hodnocení nezávislého experta:

- výhoda – přináší nový pohled na problém, který může být obrovským přínosem,

- nevýhoda – vzniká nebezpečí, že tento expert se nebude umět dobře zorientovat v prostředí znalostního systému, zvláště pokud nemá zkušenosti s vývojem takovýchto systémů.

Znalostní inženýr má jisté omezené možnosti na hodnocení systému díky tomu, že během vývoje systému už se z něj stal „skoro expert“, chybějí mu však nezbytné znalosti a zkušenosti na to, aby jeho hodnocení systému mělo stejnou váhu, jako kdyby ho vykonal expert.

V mnohých případech se expertní systémy zavádějí do organizací proto, aby se rozhodovací proces dal přesunout na nižší úrovně hierarchie. V těchto případech je účast koncového uživatele v procesu hodnocení systému irelevantní. Když je však systém navrhnutý na zlepšení kvality nebo zvýšení rychlosti rozhodování uživatele, můžou být jeho připomínky a jeho hodnocení velmi podnětné.

Způsob hodnocení systému zcela nezávislými hodnotiteli se využívá jen zřídka.

Největší problémy, se kterými se setkávají hodnotitelé při hodnocení znalostních systémů (seřaděné podle závažnosti problému) jsou:

1. Určení, zda je báze znalostí kompletní.
2. Určení, zda je báze znalostí korektní.
3. Systém neposkytuje uživateli všechny potenciální možnosti.
4. Systém se těžko používá.
5. Výsledky jsou těžko interpretovatelné.
6. Systém poskytuje uživateli nekorektní možnosti.
7. Systém nesprávně řetězí pravidla.

## 15.2 Ověřování

Ověřováním se zkoumá, zda systém splňuje specifikace, požadavky a ohraničení stanovené v předcházejících stádiích vývojového procesu. Chyby, které je možné při ověřování identifikovat, jsou *redundantnost* v bázi znalostí, *cirkularita* pravidel, *nekompletnost* báze znalostí atp. Ve snaze o systematizaci rozdělujeme tyto chyby na dvě skupiny:

1. Strukturální, interní chyby. V rámci nich rozlišujeme syntaktické chyby (např. dvě identická pravidla v bázi poznatků) a chyby, k odhalení kterých potřebujeme znát způsob inference (např. pravidlo, které nemůže být nikdy použité).
2. Funkční, externí chyby, které zahrnují situace jako redundantnost a neúplnost.

Mezi těmito dvěma třídami chyb platí, že jedna a ta stejná chyba na strukturální úrovni může způsobit několik chyb na funkční úrovni a naopak.

### 15.2.1 Redundantnost

Na problém redudantnosti můžeme nahlížet ze dvou pohledů:

1. Báze poznatků můžeme chápat jako funkci, která dává stejné výstupy pro vhodně specifikované vstupy. Můžeme zjednodušit bázi poznatků odstraněním některých pravidel tak, aby funkce zůstala stejná?
2. Když K je báze poznatků ve stavu, jak ji ve skutečnosti máme a když L je báze poznatků, jakou požadujeme. Můžeme zjednodušit K tak, že po tomto zjednodušení bude „bližší“ k L?

### 15.2.2 Nekonzistentnost

Nekonzistentnost lze interpretovat následovně:

1. Pro stejnou množinu vstupních dat expertní systém odvodí závěry, které si odporují.
2. Báze znalostí není korektní, např. když jsou porušené integritní ohraničení.
3. Jedno, nebo obě pravidla jsou nepotřebná a teda báze poznatků je redundantní.
4. Může vyjadřovat paradoxnost reálného světa – oba závěry přicházejí do úvahy a báze znalostí je v pořádku.

První dva pohledy reprezentují strukturální, další dva funkční pohled.

## Seznam doporučené literatury


[1] KELEMEN, LIDAY: Expertné systémy pre prax. Bratislava, 1996.

[2] KELEMEN, KUBÍK, LENHARČÍK, MIKULECKÝ: Tvorba expertních systémů v prostředí CLIPS. Praha, 1999.

[3] MAŘÍK: Umělá inteligence I. Praha, 1993.

[4] MAŘÍK: Umělá inteligence II. Praha, 1997.

[5] MAŘÍK: Umělá inteligence III. Praha, 2001.

[6] NOVÁK: Základy fuzzy modelování, Praha, 2002

[7] POPPER, KELEMEN: Expertné systémy. Bratislava, 1989.

[8] STEFIK: Introduction to Knowledge Systems. San Francisco, 1995.

[9] THAGARD: Úvod do kognitivní vědy. Mysl a myšlení., Praha, 2001