ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

Katedra informačního inženýrství

DIPLOMOVÁ PRÁCE

Expertní systémy v prostředí CLIPS

Vypracovala: Kristýna Procházková

Vedoucí diplomové práce: Ing. Martin Papík, Ph.D.

Zadání práce

Čestné prohlášení

Prohlašuji, že jsem tuto diplomovou práci na téma „Expertní systémy v prostředí CLIPS“ vypracovala samostatně a použila jsem pramenů, které uvádím v přiloženém seznamu literatury.

V Praze dne …….

…….
Kristýna Procházková

Poděkování

Děkuji tímto panu Ing. Martinu Papíkovi, Ph.D. za cenné připomínky a rady, odborné vedení a za čas, který mi věnoval.

EXPERTNÍ SYSTÉMY V PROSTŘEDÍ CLIPS

EXPERT SYSTEMS IN CLIPS ENVIRONMENT

Souhrn

Tato práce se zabývá jednou z nejvíce prakticky používaných oblastí umělé inteligence – expertními systémy. Jejím cílem je uvést základní informace o historii a oborech umělé inteligence, o podstatě expertních systémů, jejich vývoji, výhodách a nevýhodách jejich používání. Dále popisuje jednotlivé části expertních systémů, metody a mechanismy v nich využívané a jazyky a prostředí pro jejich vývoj a provoz.

Druhá část práce se zabývá jedním z nejoblíbenějších volně dostupných prostředí expertních systémů – CLIPS. Představuje toto prostředí z pohledu jeho historie a uvádí jeho základní vlastnosti. Obsahuje také informace o dalších prostředích a nadstavbách, které z CLIPSu vycházejí.

Praktická část práce obsahuje příklad využití expertních systémů v oblasti účetnictví a daní. Jedná se o systém pro výpočet opravných položek k pohledávkám.

Klíčová slova

Umělá inteligence, expertní systém, znalostní báze, inferenční mechanismus, báze dat, CLIPS.

Summary

This work deals with one of the most practically used domain of artificial intelligence – expert systems. Its task is to introduce the history and domains of artificial intelligence, principles and history of expert systems and the advantages and disadvantages of their use. It also describes components of expert systems, methods and mechanisms used and its languages and shells.

The second chapter deals with one of the most favourite freeware expert system shell – CLIPS. It introduces the history of this shell and its key features. It also includes information about some other shells and extensions based on CLIPS environment.

The practical part includes the example of expert systems usage in accounting and taxes. It is an expert system for doubtful receivable allowances calculation.

Keywords

Artificial Intelligence, Expert System, Knowledge Base, Inference Mechanism, Fact Base, CLIPS.

1 Úvod

Druhá polovina dvacátého století byla obdobím bouřlivého vývoje oboru nazvaného umělá inteligence. Představa stroje schopného myslet jako člověk vyvolala nadšení mezi odbornou veřejností a zpočátku vyvolala vlnu optimistických předpovědí a velkého očekávání.

V následujících desetiletích se původní cíle nepodařilo zcela naplnit, avšak do praxe byla uvedena řada řešení, která sice představovala jen částečné naplnění původních očekávání, avšak přinesla velký pokrok v mnoha oblastech vědeckého i společenského života.

Velká část těchto řešení se, často v jen mírně zdokonalené podobě, používá i ve století jednadvacátém. Jedním z takových řešení jsou expertní systémy. Tyto programové prostředky, založené na oddělení znalostní části a mechanismů, které znalosti zpracovávají, si našly široké uplatnění v mnoha oborech lidské činnosti – medicíně, průmyslu, matematice, zemědělství, vojenství, kosmonautice, vzdělávání a mnoha dalších.

Ačkoliv si to obyčejní lidé ani neuvědomují, s expertními systémy se setkávají na řadě míst v každodenním životě. Tyto programy nejen poskytují cenné rady, ale především pomáhají řešit problémové situace a v některých podobách dokonce chrání či přímo zachraňují lidské životy. Jejich pozitivní přínos pro lidstvo je nevyjádřitelný.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této práce je popsat jak umělou inteligenci obecně, tak, a to především, její nejvíce prakticky používanou oblast – expertní systémy. Práce se zaměří nejprve na historický vývoj této zajímavé oblasti, dále se bude věnovat popisu základních vlastností expertních systémů, popisu fungování jejich základních součástí a používaných mechanismů. Popíše hlavní výhody a nevýhody expertních systémů a možnosti a oblasti jejich využití. Představí také jazyky a prostředí určená pro práci expertních systémů a zaměří se na jedno z nejoblíbenějších volně dostupných prostředí – CLIPS.

V prostředí CLIPS se autorka pokusí na konkrétním příkladu z oblasti účetnictví a daní dokázat, jak může být v něm vytvořený expertní systém nápomocný v praxi.

2.2 Metodika

Podkladem pro práci je volně dostupná česká i zahraniční literatura, věnující se expertním systémům, stejně jako online dostupné vědecké články a publikace.

Pro praktickou část - práci v prostředí CLIPS je podkladem online dostupná uživatelská dokumentace. Pravidla jsou zpracována na základě zákonné úpravy dané problematiky v Zákoně č. 593/1992 Sb., o rezervách pro zjištění základu daně z příjmů. Praktický příklad je vypracován v CLIPS verze 6.3 pro MS Windows. Samotný kód je psán v poznámkovém bloku MS Windows.

3 Expertní systémy

3.1 Umělá inteligence

Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který - kdyby ho dělal člověk - bychom považovali za projev jeho inteligence [1].

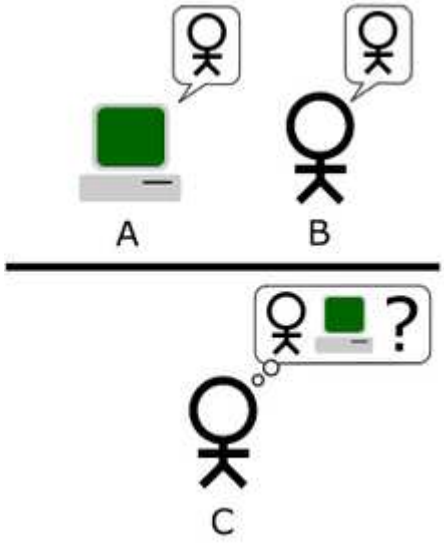
Umělá inteligence se zabývá tím, jak počítačově řešit úlohy, které dnes zatím zvládají lidé lépe [2].

Norma ISO 2382-28:1988 definuje umělou inteligenci jako "schopnost stroje simulovat myšlenkové pochody člověka tím, že vykonává činnosti obvykle spojované s lidskou inteligencí, např. uvažování, učení a sebezdokonalování" [3].

Pojem umělá inteligence (aftificial intelligence) použil jako první americký informatik a pozdější zakladatel jazyka LISP John McCarthy na konferenci v Darthmouth College ve státě New Hampshire v USA. Cílem konference s názvem The Dartmouth summer research project on artificial intelligence bylo dokázat, že každý aspekt učení nebo jakýkoliv jiný příznak lidské inteligence se dá v principu tak přesně popsat, že může být vyvinut stroj, na němž je možné jej simulovat.

Na konferenci dokonce autoři Allen Newell, J.C. Shaw and Herbert Simon představili první funkční program v oblasti umělé inteligence - Logic Theorist (LT). Optimismus přítomných na této konferenci vedl k předpovědi, že do deseti let bude počítač schopen zvítězit na mistrovství světa v šachu, objevit a dokázat nový matematický teorém, a bude schopen porozumět přirozenému jazyku a bude sloužit jako překladatel.

Otázkou, zda stroje dokáží myslet, se ale lidstvo zabývalo již dříve. Již v roce 1950 sepsal významný britský matematik, logik, kryptoanalytik a zakladatel moderní informatiky Alan Turing článek nazvaný Computing Machinery and Intelligence test, který začíná slovy: „Navrhuji položit si otázku – Mohou stroje myslet?“ [4], v němž se zabýval možností, jak zjistit, zda se libovolný systém umělé inteligence chová jako skutečná inteligentní entita (člověk). Podle Alana Turinga byl pojmenován tzv. Turingův test, jehož princip probíhá podle obrázku 1.



Obr. 1 – Princip Turingova testu [1]

Systém umělé inteligence (A) a reálná inteligentní entita (B) se nachází na místě odděleném od testující osoby (C). Tato osoba pokládá v přirozeném jazyce otázky, na něž náhodně odpovídá buď systém umělé inteligence, nebo skutečná osoba. Pokud testující osoba nedokáže z odpovědí poznat, zda autorem odpovědi je systém nebo člověk, splňuje systém umělé inteligence Tuningův test. O takovém systému tedy lze říci, že se chová inteligentně.

Proti této koncepci se ozvala řada námitek, mezi ty známé patří například tzv. Paradox čínského pokoje od amerického filozofa Johna Searlea. Ten namítá, že i člověk naprosto neznalý čínštiny může smysluplně třídit tabulky s čínskými znaky, stačí mu pouze vědět, kde a jak hledat správné řešení [5]. Searle tím prokazoval, že pro splnění Turingova testu ještě není třeba nic chápat, stačí pouze imitovat.

Příkladem, jak Turingovým testem může úspěšně projít i neinteligentní entita, je program ELIZA, který v roce 1966 představil informatik Joseph Weizenbaum. Tento jednoduchý program, který imitoval rozhovor pacienta a psychologa, dokázal na základě využití klíčových slov ze vstupů od uživatele najít předem definovanou odpověď. Mnoho uživatelů uvěřilo, že jejich partnerem v dialogu je člověk.

I přes řadu námitek je princip Turingova testu i po šedesáti letech od jeho publikování prakticky využíván například v prostředí internetu (systém CAPTCHA).

V padesátých letech minulého století se zformovala centra výzkumu umělé inteligence, a to na Massachusettském technologickém institutu a na univerzitě Carnegie Mellon. Výzkumníci v roce 1957 představili první verzi programu GPS – General Problem Solver, určeného pro efektivní řešení problémů.

Významným rokem byl rok 1958, kdy John McCarthy vyvinul programovací jazyk LISP. Ačkoliv původním cílem McCarthyho nebylo vytvořit nový jazyk, ale pouze doplnit v té době používaný jazyk Fortran o možnost práce se seznamy (List Processing), představil silný programovací nástroj, který se již od svého uvedení začal používat pro účely výzkumu umělé inteligence a používá se dodnes (například jako vnitřní programovací jazyk v AutoCADu).

Významným počinem ve vývoji umělé inteligence bylo formulování rezolučního principu v roce 1965 filozofem a matematikem Johnem Alanem Robinsonem. Tento postup dokazování platností logických formulí umožnil rozvoj v oblasti automatických odvozovacích systémů.

V roce 1972 vytvořili Alain Colmeraurer a Robert Kowalski programovací jazyk Prolog. Ten představoval významnou změnu, protože oproti tradičnímu strukturovanému programování umožnil řešit úlohy logickým popisem úlohy. Tento jazyk si získal velkou popularitu a používá se dodnes.

Již v průběhu šedesátých let začalo být jasné, že původní cíle, které si vědci uložili splnit do roku 1970 (počítač bude velmistrem v šachu, odhalí nové matematické teorémy, porozumí přirozenému jazyku a bude sloužit jako překladatel, a že bude schopen skládat hudbu na úrovni klasiků [6]), nebude možné dosáhnout. Snaha najít univerzální řešící postup byla postupně redukována. Bylo zjištěno, že stěžejním faktorem pro aplikace umělé inteligence jsou znalosti – jejich získávání, kvalita, rozsah a prezentace. Obecný formální aparát pro řešení úloh poskytuje pouze nástroj pro využívání těchto znalostí a hraje druhořadou roli. V období 70. – 90. let minulého století se právě reprezentace znalostí dostala do popředí výzkumu umělé inteligence. V této době se objevují první expertní systémy.

Kromě expertních systémů, kterými se zabývá tato práce, existují další aplikační oblasti, které se v současné době v umělé inteligenci rozvíjejí. Jedná se například o:

a) neuronové sítě,
b) robotiku,
c) zpracování přirozeného jazyka,
d) hraní her,
e) dokazování teorémů,
f) rozpoznávání obrazů,
g) umělý život,
h) počítačové vidění,
i) strojové učení,
j) dobývání znalostí z databází,
k) multiagentní systémy,

a další [7].

3.2 Expertní systém

Expertní systémy jsou prakticky používanou aplikační oblastí umělé inteligence. Existuje značné množství definic expertních systémů, například:

a) Expertní systémy jsou programové prostředky určené k řešení takových úloh, které jsou považovány za obtížné a jejichž uspokojivé řešení může provést pouze specialista v daném oboru – expert [8]

b) Expertní systémy jsou programy, které využívají vhodně v počítači uložených poznatků lidských expertů k řešení problémů, které obvykle v praxi vyžadují znalost expertů [9]

c) Expertní systémy jsou systémy podporující uživatele při řešení problémů, které vyžadují znalosti experta o jisté oblasti.

d) Expertní systém je heuristický systém, který umožňuje řešení problémů v dané aplikační oblasti vyvozováním závěrů z báze znalostí získané lidskou zkušeností. [3]

Ačkoliv žádná z definic ještě nebyla ustanovena jako oficiální, nejčastěji uváděnou je následující definice:

„Expertní systémy jsou počítačové programy, simulující rozhodovací činnost experta při řešení složitých úloh a využívající vhodně zakódovaných, explicitně vyjádřených speciálních znalostí, převzatých od experta, s cílem dosáhnout ve zvolené problémové oblasti kvality rozhodování na úrovni experta.“ [11]

V souvislosti s expertními systémy se lze setkat také s obecnějším termínem znalostní systém (knowledge-based system). Tento pojem zahrnuje také systémy, které neobsahují znalosti na úrovni experta, ale v současnosti jsou termíny znalostní systém a expertní systém často používány jako synonyma.

3.3 Historie expertních systémů

- 50. – 60. léta
  - GPS – General-purpose Problem Solver (GPS) – snaha autorů Herberta Simona, J.C. Shawa , and Allena Newella o univerzální systém, řešící dané problémy, považován za předchůdce expertních systémů
- 60. léta
  - výzkumníci zjišťují, že:
    - mechanismus řešící problémy je jen malá část kompletního, inteligentního počítačového systému
    - GPS pro vybudování výkonného expertního systému nestačí
    - expertní systémy musí být neustále doplňovány novými informacemi
    - složitost problémů vyžaduje značné množství znalostí o řešené oblasti
  - DENDRAL: jeho úkolem je pomáhat identifikovat chemické sloučeniny na základě spektrografických dat. Jedná se o plánovací systém, který je určen

k odvozování struktur chemických látek na základě histogramů rozložení hmotností získaných ze spektrometru.

  - MACSYMA: poskytuje soubor nástrojů pro manipulaci s matematickými výrazy a vzorci. Je využíván např. nukleárními fyziky, kteří jsou často nuceni řešit soustavy velkého počtu rovnic.

- 70. léta
  - představení mnoha skutečných expertních systémů
  - zjištění, že stěžejní roli hrají znalosti, síla expertních systémů spočívá právě ve znalostech
  - vědci v oblasti umělé inteligence vyvíjejí
    - komplexní teorie o reprezentaci znalostí
    - univerzální rozhodovací procedury a inferenční mechanismy
  - omezený úspěch - znalosti jsou příliš rozsáhlé a různorodé
  - MYCIN: na základě dostupných dat rychle určoval typy bakteriální infekce, kterými by mohl být nově hospitalizovaný postižen a navrhovat vhodnou léčbu antibiotiky tak, aby se stav pacienta stabilizoval do doby, než jsou dokončena časově náročná laboratorní vyšetření.
  - PROSPECTOR: slouží k vyhodnocování geologických dat s cílem rychle rozhodnout, zda v dané lokalitě provádět podstatně dražší hloubkové vrty
  - HEARSAY II.: první systém, který ukázal, že počítač by mohl v budoucnosti spolehlivě a rychle rozumět přirozené řeči v úzce vymezené předmětné oblasti
  - PUFF: poskytování konzultací ohledně potíží s dýchacími cestami
  - INTERNIST: považován za jeden z nejrozsáhlejších expertních systémů v historii, později byl přejmenován na systém CADUCEUS, obsahující údajně 85% veškerých znalostí z interního lékařství. Oba systémy se i dnes využívají v lékařské praxi.
- 80. léta - současnost
  - expertní systémy se stávají komerčně dostupnými
    - XCON
    - XSEL
    - CATS-1
  - nástup programovacích nástrojů a shellů

- EMYCIN
- EXPERT
- META-DENDRAL
- EURISKO

3.4 Aplikační oblasti a příklady expertních systémů

oblast	**příklady systémů**
geologie	DIPMETER ADVISOR, FEE Tool, FINDER, GEOPLAY, muPETROL, PROSPECTOR, X-net, XUMA, LITHO, MUD
chemie	CRYSALIS, DENDRAL, META-DENDRAL, SECS, SYNCHEM, TQMSTUNE, CLONER, MOLGEN, SPEX
medicína	CADIAG, CASNET/GLAUCOMA, DM, EEG, HEADMED, INTERNIST/CADUCEUS, MEDICO, MYCIN, ONCOCIN, PIP, PROTIS, PUFF, RHEUM, VM, ABEL, COAG, ANNA, BLUE BOX, ATTENDING, GUIDON
počítačové systémy	PTRANS, BDS, XCON, XSEL, XSITE, YES/MVS, TIMM
průmysl	MECHANO, REACTOR, SACON, DELTA, STEAMER
zemědělství	PROPLANT, Rice-Crop Doctor, AGREX, CALEX, VARIEX
matematika	AM, EURISKO, MACSYMA, QUIST, MATHPERT
právo	LEGOL, LRS, MATRIM, TAXMAN, LEX
elektronika	ACE, IN-ATE, NDS, EURISKO, PALLADIO, REDESIGN, CADHELP, SOPHIE
vojenství, policie	STRATEGY 1, GE/DARPA, MBEES, PROMIS, REBES, TrueAllele
letectví, kosmonautika	TRANS, SPIKE, AIRPLANE, Expert Navigator, NAVEX, HEAT, EXIMU
účetnictví	TAXMAN, CAPEX, TICOM, S.C.A.C.CO.
výuka	BLAH, GUIDON, SOPHIE, WHY

3.5 Charakteristika expertních systémů

Podle [12] jsou expertní systémy charakterizovány těmito rysy:

e) oddělení znalostí a mechanismu pro jejich využívání

Znalosti experta jsou uloženy v bázi znalostí odděleně od inferenčního (odvozovacího) mechanismu na rozdíl od konvenčních programů, kde jsou znalosti experta pevně zakódovány v jednotlivých instrukcích programu, které se aplikují v předem stanoveném pořadí. To umožňuje vytvářet problémově nezávislé (prázdné) expertní systémy, ve kterých jeden inferenční mechanismus může pracovat s různými bázemi znalostí.

(viz také kapitola Struktura expertního systému)

f) neurčitost v bázi dat

V bázi znalostí jsou uloženy nejen exaktně dokázané znalosti, ale i zkušenosti experta, které se mu osvědčily v jeho praxi. Mohou se zde pak objevit pojmy jako "většinou", "často", "nevím" apod. Tyto pojmy je nutno kvantifikovat (viz také kapitola Neurčitost v expertních systémech).

g) neurčitost v datech

Konkrétní data o řešeném případu bývají zatížena neurčitostí způsobenou nepřesně určenými hodnotami nebo subjektivním pohledem uživatele (viz také kapitola Neurčitost v expertních systémech).

h) dialogový režim

Expertní systémy jsou obvykle vytvářeny jako tzv. konzultační systémy. Uživatel se systémem komunikuje způsobem "dotaz systému - odpověď uživatele" (podobně by komunikoval i s lidským expertem). Z hlediska vnějšího chování je na expertní systémy kladena řada požadavků odvíjejících se z představy, že expertní systém nahrazuje odborníka, poskytujícího konzultaci laikovi či méně zkušenému odborníkovi.

i) vysvětlovací činnost

Expertní systém by měl poskytnout vysvětlení svého uvažování, aby se zvýšila důvěra uživatelů v přijaté závěry a doporučení. Počítačový expertní systém musí být nejen schopen vést s uživatelem dialog ve formě otázka-odpověď (viz bod d), nýbrž musí být schopen vysvětlit a zdůvodnit dílčí závěry i položit vhodný doplňující dotaz stejně tak, jako být připraven zaměřit své úsilí na řešení podproblémů specifikovaných uživatelem v průběhu konzultace (viz také kapitola Požadavky na expertní systém).

j) modularita a transparentnost báze znalostí

Pro účinnost expertního systému je rozhodující báze znalostí. Znalosti experta nemají statický charakter, nýbrž se postupně vyvíjejí a rozrůstají. Modularita umožňuje snadnou aktualizaci báze znalostí. Báze znalostí musí být též transparentní (tedy čitelná a srozumitelná) jak pro experta (aby ji mohl upravovat a rozšiřovat), tak i pro další odborné pracovníky, kteří z ní mohou čerpat potřebné znalosti. Vytváření báze znalostí probíhá interaktivně konzultacemi experta z dané problémové oblasti s odborníkem na tvorbu bází (znalostním inženýrem). Báze znalostí je tak postupně upravována, až chování expertního systému odpovídá představám experta.

3.6 Požadavky na expertní systém

a) vysoký výkon

Systém musí být schopen odpovídat na stejné nebo i vyšší kvalitativní úrovni než lidský expert [13].

b) přiměřený čas odezvy

Systém musí reagovat v přiměřeném čase, srovnatelném, nebo lepším než v případě lidského experta. Důraz na časové omezení je obzvláště významný v případě expertních systémů pracujících v reálném

čase (real-time expert systems), kde je čas odezvy v požadovaném časovém intervalu klíčovou záležitostí [13].

c) důvěryhodnost

Výsledky systému musí být důvěryhodné a systém nesmí být náchylný k chybám [13].

d) srozumitelnost – schopnost vysvětlování

Systém by měl být schopen srozumitelné doložit jednotlivé kroky svého usuzovacího procesu stejně, jako to dokáže lidský expert. Neměl by fungovat jako černá skříňka, která poskytují jen nezdůvodněné výstupy. Jedním z důvodů, proč je schopnost vysvětlování důležitá, je, že výsledky expertízy mohou významně ovlivňovat lidské životy a majetky. Následky chybného nebo nepřesného rozhodnutí by mohly být závažné.

Druhým důvodem je, že tato schopnost umožňuje ověřit správnost procesu získávání znalostí v průběhu tvorby expertního systému a schopnost systému znalosti správně využívat. V průběhu tvorby znalostní báze může docházet k nechtěným chybám, překlepům či nedorozuměním mezi expertem a znalostním inženýrem. Jiným zdrojem chyb mohou být také nepředvídatelné interakce v rámci systému.

Protože proces rozhodování expertního systému není sekvenční, jako je tomu u konvenčního programování, nelze pro pochopení jeho usuzování procházet pravidla postupně tak, jak byla do systému zadávána. Právě dobrá vysvětlovací schopnost systému umožňuje ověřit jak správnost znalostní báze, tak i správnost procesu usuzování [13].

e) přizpůsobivost

vzhledem k velkému množství znalostí, kterými expertní systémy disponují, je důležité mít účinný mechanismus pro přidávání, modifikaci a mazání v bázi znalostí.

3.7 Výhody expertních systémů

Podle [13] jsou hlavní výhody expertních systémů následující:

a) vyšší dostupnost - expertíza je dostupná na širokém okruhu hardware a software

b) nízká cena – cena expertízy v přepočtu na uživatele je nižší, než v případě lidského experta

c) nízké riziko – expertní systém může být použit i v prostředí, které není pro lidského experta bezpečné

d) stálá dostupnost – expertíza je na rozdíl od lidského experta trvale k dispozici

e) vícenásobná expertíza – v jakémkoliv okamžiku mohou být současně k dispozici znalosti různorodých expertů

f) vysoká důvěryhodnost – expertní systém dokáže eliminovat neshody a konflikty v rozhodování více lidských expertů

g) vysvětlovací schopnost – expertní systém bývá schopný zpětně doložit postup svého usuzovacího procesu, k čemuž nemusí být lidský expert, ať již z důvodů profesních, časových či osobních, vždy ochotný

h) rychlá odezva – expertní systém je schopen v závislosti na použitých systémových prostředcích odpovědět stejně rychle či ještě rychleji než lidský expert

i) vždy stabilní, neemotivní a kompletní expertíza – expertní systémy nejsou jako lidský expert ovlivňovány například stresem nebo únavou

j) inteligentní učeni – expertní systémy mohou díky vysvětlovací schopnosti sloužit jako výuková pomůcka

3.8 Nevýhody expertních systémů

Mezi nevýhody expertních systémů patří skutečnost, že k předání znalostí systému většinou nestačí sám expert, ale musí být přítomen znalostní inženýr. Vysoká specializace znalostního inženýra a délka procesu budování znalostní báze má za následek vyšší finanční náročnost projektu, než kdyby mohl expert své znalosti zadávat do báze sám. Toto sice je technicky možné například v případě prázdných expertních systémů, nicméně expert v určité specializované oblasti jen málokdy bývá schopen pracovat s expertními systémy.

Přítomnost znalostního inženýra a tím způsobené zprostředkované předávání znalostí také zvyšuje riziko nepřesnosti informací a možnost nedorozumění.

Další nevýhodou je, že lidský expert často nedokáže podrobně a úplně definovat a popsat své rozhodovací procesy, stejně tak jako nemožnost promítnout do expertního systému zdravý rozum či intuici, které často mají podstatný vliv na rozhodnutí.

V úvahu je třeba vzít také skutečnost, že zatímco znalosti a dovednosti člověka se s časem a získanými zkušenostmi vyvíjejí, expertní nebo znalostní systém bude bez lidského zásahu i za dvacet let navrhovat stejné postupy [15].

3.9 Vhodnost a nevhodnost použití expertního systému

Expertní systémy představují vhodný nástroj při hledání řešení v mnoha oblastech. Dokáží řešit různé typy úloh, jako například:

a) diagnostika
b) návrh a konfigurace
c) vyučovací proces
d) interpretace
e) monitorování
f) plánování
g) prognostika

h) řízení

i) simulace

j) selekce

a další.

Před samotným nasazením expertního systému je potřebné zvážit, zda právě toto řešení je tím nejvhodnějším, a zda neexistuje vhodnější paradigma, například konvenční programování. Nejvhodnější jsou expertní systémy v případech, kdy informace, které jsou k dispozici, jsou převážně heuristické a nejisté, stejně tak jako situace, kdy lidský expert hledá řešení metodou pokus-omyl a kdy není možné sestavit klasický algoritmus řešení problému.

Budovat expertní systém má smysl pouze tehdy, je-li k dispozici expert ochotný a schopný dát k dispozici své znalosti, stejně jako musí být přítomna ochota a chuť zaplatit, používat a nadále podporovat expertní systém na straně zadavatele řešení.

3.10 Typologie expertních systémů

a) z hlediska formy reprezentace znalostí (více viz kapitola „Struktura expertních systémů“)

   - produkční (pravidlové) systémy
   - založené na matematické logice
   - sémantické sítě
   - rozhodovací stromy
   - rámce (objekty)

b) z hlediska charakteru řešených úloh

   - diagnostické expertní systémy - Úkolem těchto systémů je porovnávat a vyhodnocovat předem stanovené hypotézy. Cílem je určit, která z těchto hypotéz nejlépe odpovídá reálným datům. [14]

- plánovací (generativní) expertní systémy - Při řešení úloh v plánovacím expertním systému je znám cíl řešení a počáteční stav. Úkolem systému je s využitím dat o daném případu nalézt optimální posloupnost kroků (operátorů), kterými lze dosáhnout stanoveného cíle. Výsledkem je seznam navrhovaných řešení, která jsou ohodnocena určitou měrou optimality. Důležitou částí je zde generátor možných řešení, který automaticky generuje, kombinuje a testuje přípustná řešení. Vybraná řešení jsou testována na datech z báze dat. [14]

- hybridní systémy – Kombinací architektury diagnostického a plánovacího systému vzniká hybridní expertní systém. Tímto typem jsou například inteligentní výukové systémy či monitorovací systémy. [14] Příkladem jsou monitorovací expertní systémy, které v případě diagnostiky určité události aktivují subsystém pro plánování zásahu.

c) z hlediska obecnosti

- prázdný expertní systém – Expertní systém bez problémově závislých částí (tj. bez báze znalostí a báze dat). Tento typ expertních systémů se podařilo vyvinout pouze pro řešení diagnostických úloh (diagnostické expertní systémy). Plánovací a hybridní expertní systémy mají totiž výrazně problémově závislou bázi znalostí. [14]

- problémově orientovaný expertní systém – Prázdný expertní systém doplněný o bázi znalostí. Je použitelný k řešení úloh pouze v určité problémové oblasti (architektura, typ reprezentace znalostí a řídicí mechanismus jsou těsně spojeny s danou oblastí) [14]

- hotový expertní systém – obsahuje všechny části – uživatelské rozhraní, bázi dat, bázi znalostí a inferenční mechanismus – a je určen k okamžitému použití při řešení konkrétního problému.

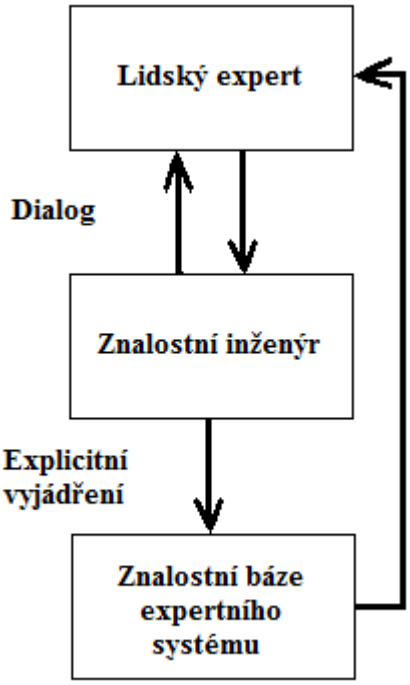
- expertní systémy pracující v reálném čase

3.11 Struktura expertních systémů

3.11.1 Znalostní báze

Obsahuje znalosti o problémové oblasti a zahrnuje fakta, pravidla, koncepce a vztahy. Tvorba znalostní báze je u větších systémů výsledkem spolupráce znalostního inženýra a experta, jak již bylo popsáno dříve.

Znalosti v bázi mají různou úroveň obecnosti – od zcela obecných a všeobecně známých, až po vysoce specifické, poskytnuté doménovým expertem. Princip tvorby znalostní báze je ilustrován na obrázku 2.



Obr. 2 – Vývoj expertního systému [2]

Báze by měla splňovat požadavek modularity, kdy cílem je zachovat přehlednost báze pro její případné budoucí doplňování. Měla by být rovněž strukturalizována, tedy měla by být vytvořena hierarchie pojmů, které se v bázi vyskytují. Znalosti bývají nejčastěji reprezentovány formou:

a) produkčních pravidel

Jedná se o nejrozšířenější formu reprezentace znalostí v expertních systémech. Produkční pravidlo sestává ze dvou částí – levá strana obsahuje podmínku a nazývá se antecendent, pravá strana obsahuje závěry či akce platné v případě splnění podmínky a nazývá se konsekvent. Příklady pravidel:

1) IF žena má dítě THEN žena je matka

2) IF auto nestartuje AND světla nesvítí THEN baterie je vybitá

3) IF rychlost je vysoká AND prší AND NOT stěrače stírají THEN zapni stěrače AND sniž rychlost

Ze druhého a třetího příkladu je vidět, že antecendent může obsahovat kromě jednoduchých podmínek také několik tvrzení, jejichž individuální platnost je zkombinována pomocí logických spojek AND, OR a NOT. V případě systémů nepodporujících neurčitost se nejčastěji pracuje s datovými typy číslo (je-li x 20), nebo znakovými řetězci (jméno je „Karel“).

Systémy podporující neurčitost umožňují zpracovat další datové typy – fuzzy čísla (je-li věk přibližně 30), a fuzzy množiny (např. je-li rychlost vysoká, kde rychlost může být nízká, střední či vysoká). Logické spojky lze využít také v konsekventu.

b) matematické logiky

Znalosti jsou ve znalostí bázi reprezentovány pomocí logických formulí. Výroková logika má význam pro inferenční mechanismy expertních systémů. Umožňuje logické operace nad výroky. Zahrnuje tyto operace – negaci, konjunkci, disjunkci, implikaci a ekvivalenci. Predikátová logika je bohatší než výroková. Obsahuje navíc například univerzální a existenční kvantifikátory. Na predikátové

logice jsou založeny některé programovací jazyky expertních systémů, například PROLOG.

c) sémantických sítí

Znalosti jsou reprezentovány pomocí sítí, založených na obecných sítích, představující orientované grafy. Uzly zde představují jednotlivé objekty, hrany grafu představují relace mezi uzly.

Obecné sítě však nejsou dostačující, protože nepopisují míru závislostí, tedy dostatečně nepopisují, o jaký vztah mezi uzly se jedná. Toto umožňuje sémantická síť. Jedná se o ohodnocený orientovaný graf. Podporuje dědičnost a tranzitivitu.

d) rozhodovacích stromů

Rozhodovací stromy vycházejí z předpokladu, že menší problémy se zvládají snáze, než velké. Je tedy založen na metodě „rozděl a panuj“. Znalosti se postupně rozdělují na menší a menší podmnožiny tak, aby v těchto podmnožinách převládaly znalosti jedné třídy. Od kořene stromu se na základě odpovědí na otázky (umístěné v nelistových uzlech) postupuje příslušnou větví stále hlouběji, až do listového uzlu, který odpovídá zařazení znalosti do třídy.

e) rámců (objektů)

Rámce jsou datové struktury, jejichž atributy jsou všechny znalosti o dané konkrétní entitě – objektu. Kromě atributů mohou objekty obsahovat také procedury – metody, které je používají. Rámce umožňují dědění od objektů na vyšší hierarchické úrovni.

3.11.2 Inferenční (odvozovací, usuzovací) mechanismus

Jedná se o řídící mechanismy, jádro expertního systému, které realizuje proces hledání řešení nad bází znalostí a vstupními informacemi od uživatele.

Představují procedurální složku expertního systému. Jsou analogií k rozhodovacímu procesu lidského experta.

Inferenční mechanismus pracuje v následujících krocích

a) inicializace báze faktů – dosazení „startovacích položek“

b) spuštění zvoleného usuzovacího algoritmu

c) ukončení běhu v případě, že již není k dispozici žádné další pravidlo, nebo pokud byl běh procesu ukončen přímo instrukcí v pravé straně spuštěného pravidla.

Při usuzování lze zvolit jako odvozovací techniku:

a) dopředné řetězení (forward chaining), neboli usuzování řízené daty

Využívá pravidla modus ponens, představující přímé usuzování, které říká, že platí-li předpoklad E a pravidlo E→H, pak platí závěr H. Dopředné řetězení je vhodné pro problémy zahrnující plánování, monitorování a řízení. Představuje rozhodování směrem od faktů k hypotézám. Dopředné řetězení je inferenčním mechanismem mnoha expertních systémů, jako například CLIPS, ILOG-RULES či OPS5.

Proces probíhá v následujících krocích:

1) Položky z báze faktů jsou porovnávány s levou stranou pravidel z báze znalostí.

2) Při nalezení více než jednoho vyhovujícího pravidla se rozhoduje, které se vybere – dochází k řešení konfliktů. Nejčastěji používané způsoby řešení konfliktů jsou

   - Pravidlo nesmí být spuštěno víckrát se stejnými daty.

   - Systém přednostně využívá čerstvější data – k položkám v bázi faktů je přiřazen časový údaj, odpovídající okamžiku, kdy byly do báze zařazeny. Někdy bývají naopak upřednostňována nejstarší data.

- Komplexnější pravidla s více podmínkami mají přednost před jednoduššími pravidly.

3) Aplikuje se vybrané pravidlo a následně se přidává nová položka do báze faktů, případně se vymaže existující položka a proces se vrací ke kroku 1 [15].

V systémech využívajících mechanismus dopředného řetězení bývá často zahrnut také usuzovací mechanismus RETE. Tento mechanismus byl vyvinut s cílem odstranit velkou nevýhodu dopředného řetězení – nutnost v každém cyklu opakovaně porovnávat všechna pravidla se všemi fakty, způsobující neefektivnost tohoto usuzovacího procesu. Algoritmus RETE využívá síťovou strukturu a vychází ze skutečnosti, že spuštění pravidla většinou mění pouze několik málo faktů a jen málo pravidel je ovlivněno každou takovou změnou. Rovněž využívá faktu, že na levých stranách pravidel se často objevují shodné podmínky. Nevýhodou algoritmu RETE je vyšší nárok na paměť [17]. Tento algoritmus je využit například v systémech CLIPS nebo OPS5.

b) zpětné řetězení (backward chaining), neboli usuzování řízené cíly

Je vhodné pro problémy zahrnující diagnostiku, které mají malý počet cílových hypotéz. Představuje usuzování probíhající směrem od hypotézy k faktům. Báze faktů v tomto případě neobsahuje odvozená fakta, ale cíle, případně podcíle. Mechanismus vychází z konsekventů pravidel a zjišťuje, jsou-li splněny jejich předpoklady. Systém nejprve vytvoří seznam pravidel, jejichž splněním je možné dosáhnout cíle. Předpokladové části těchto pravidel se stávají podcíly a celý proces se opakuje [15].

Proces probíhá v následujících krocích:

1) Vytvoří se zásobník naplněný všemi cíli.

2) Shromáždí se všechna pravidla schopná splnit cíl na vrcholu zásobníku. V případě prázdného zásobníku proces končí.

3) Postupně se zkoumají pravidla z předchozího kroku následovně:

- Je-li splněna levá strana pravidla, provádí se pravá strana. Byl-li zkoumaný cíl koncový, pak se odstraní ze zásobníku a pokračuje se krokem 2. Jednalo-li se o podcíl, odstraní se ze zásobníku a pokračuje se zpracováním dočasně odloženého předchozího pravidla.

- Nesplňují-li cíle z báze faktů podmínky pravidla, zkoumání pravidla je ukončeno.

- Chybí-li v bázi faktů hodnota pro vyhodnocení podmínky pravidla, hledá se pravidlo, z nějž by mohla být odvozena. Podaří-li se to, vloží se parametr do zásobníku jako podcíl, zkoumané pravidlo se dočasně odloží a přejde se na krok 2. V opačném případě se chybějící hodnota zjistí od uživatele a pokračuje se krokem 3a) zkoumáním další podmínky.

4) Není-li možné pomocí žádného pravidla odvodit hodnotu důsledku, zůstává daný cíl neurčen, odstraní se ze zásobníku a pokračuje se krokem číslo 2.

V následující tabulce jsou uvedeny některé charakteristiky dopředného a zpětného řetězení:

Dopředné řetězení	**Zpětné netěžení**
Plánování, monitorování, řízení	Diagnostika
Od přítomnosti k budoucnosti	Od budoucnosti k přítomnosti
Od předcházejícího k následnému	Od následujícího k předcházejícímu
Řízení daty, usuzování zdola nahoru	Řízení cílem, usuzování shora dolů
Postup dopředu k nalezení řešení	Postup zpět k nalezení faktů, které

plynoucího z faktů	podporují hypotézu
Užití hledání do šířky	Užití hledání do hloubky

Tab. 1 – Některé charakteristiky dopředného a zpětného řetězení [1]

Zatímco některé expertní systémy zahrnují jen jeden z popsaných inferenčních mechanismů, jiné nabízejí nejen obě metody, ale umožňují také obousměrnou inferenci prostřednictvím možnosti přepínání mezi metodami v průběhu procesu.

Především u rozsáhlých systémů je výhodou možnost definovat tzv. agendy. Jedná se o posloupnosti úkolů, které systém provádí. Spuštění jednotlivých úkolů může být podmíněno na základě předchozích výsledků usuzovacích procesů.

3.11.3 Báze faktů (dat)

Báze faktů představuje obsah pracovní paměti při práci expertního systému. Zahrnuje konkrétně zadané či odvozené fakty. Na počátku práce inferenčního mechanismu se zde vkládají vstupní údaje, v průběhu usuzovacího procesu se zde ukládají odvozená fakta, mezivýsledky a výsledky. Fakty mají poměrně jednoduchou strukturu, během inferenčního procesu se zpravidla mění a přistupuje se k nim s velkou frekvencí

3.11.4 Další komponenty expertních systémů

d) uživatelské rozhraní

Umožňuje komunikaci expertního systému s uživatelem, znalostním či vývojovým inženýrem, a to v průběhu získávání dat, definování pravidel a faktů a v průběhu rozhodovacího procesu. Zajišťuje také prezentaci výsledků uživateli. Může být založeno na textové formě, avšak častější a uživatelsky příjemnější jsou grafická rozhraní.

e) rozhraní pro spolupráci s externími systémy

Na expertní systém nelze nahlížet jako na izolovaný systém. Kromě kontaktu s lidmi bývá velmi často propojen i s dalšími systémy, například databázemi či procedurálními programy. Systém využívá externí systémy pro získávání dat, která sám nemá k dispozici.

Rozhraní umožňuje spolupracovat s externím zdrojem jak z podnětu samotného systému například při získávání faktů či pravidel, tak i z podnětu externího systému, který může využívat expertní systém například pro interpretaci dat nebo pro výběr vhodného postupu pro další činnost.

f) vysvětlovací mechanismus

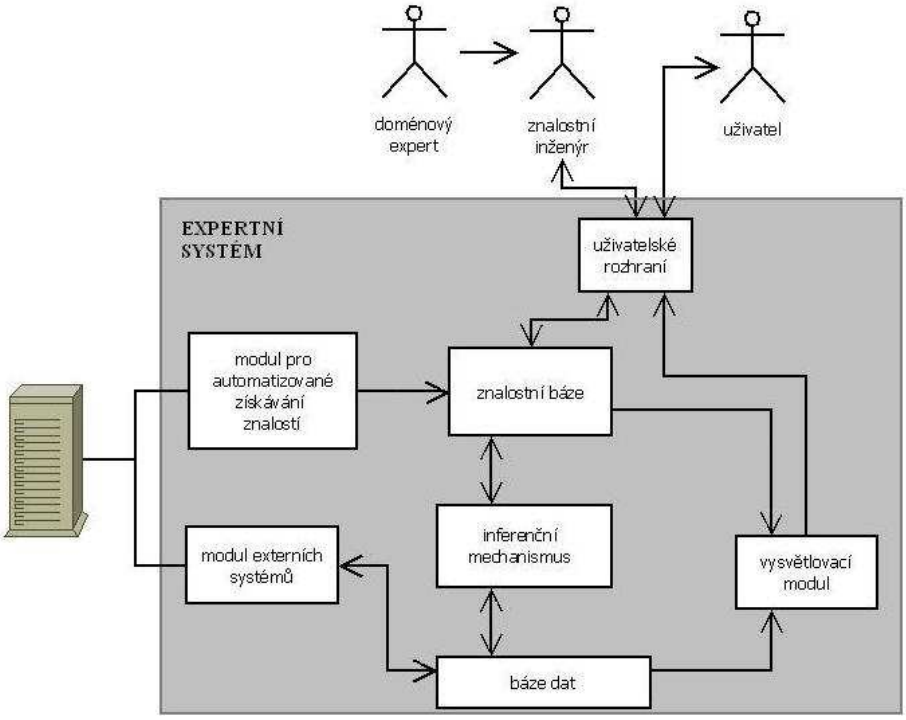
Umožňuje průhlednost usuzovacího procesu. Uživatel se zde může dozvědět, proč mu byla položena konkrétní otázka, nebo jak systém dospěl k určitému závěru. Není součástí všech expertních systémů. Je důležitý nejen v průběhu samotného používání expertního systému, ale také v průběhu jeho tvorby. Ze své podstaty rovněž umožňuje uživateli možnost učení.

g) vývojový modul

Různá prostředí expertních systémů umožňují vývojářům rozdílné možnosti pro vývoj a vylepšování systému. Narozdíl od jednoduchých systémů dovolují ty více sofistikované širší možnosti reprezentace znalostí, inferenčních technik a designu uživatelského rozhraní, stejně jako i ladění systému. Rovněž možnost on-line nápovědy je velmi užitečnou vlastností některých systémů.

h) modul pro automatizované získávání znalostí

Umožňuje získávat znalosti i jinak, než přímo od doménového experta. Jedním ze zdrojů jsou učící se systémy, jiným například systémy pro dolování dat (data-mining).



Obr. 3 – Struktura expertního systému

Jak již bylo zmíněno, ne každý expertní systém obsahuje všechny výše uvedené součásti. Například tzv. prázdné expertní systémy obsahují pouze uživatelské rozhraní a inferenční mechanismus, případně i mechanismus vysvětlovací. Bázi znalostí si vytváří sám uživatel (např. EXSYS, KAS, VP Expert, Mentor, EMYCIN, AL/X, CLIPS, NEST a další).

3.12 Jazyky expertních systémů

Jak již bylo zmíněno dříve, klasické procedurální programování není pro účely expertních systémů vhodným řešením. Pro tyto jazyky je typické velmi úzké propojení datové struktury a prostředků pro manipulaci s nimi. Klasický procedurální program je tvořen posloupností příkazů a obsahuje algoritmus, jak danou úlohu řešit. Data jsou v tomto případě velmi úzce propojena s prostředky pro práci s nimi.

Naproti tomu jazyky expertních systémů umožňují dvě úrovně abstrakce – datovou abstrakci a znalostní abstrakci. Typickým znakem jazyků expertních systémů je oddělení dat od metod pro manipulaci s nimi [13].

Výhodou jazyků expertních systémů je jejich relativní jednoduchost, nevýhodou je ale omezený rozsah problémů, které jimi jdou řešit. I z toho důvodu je často využívána možnost přecházet dočasně na vykonání instrukcí psaných v procedurálních jazycích.

Příklady prostředí dostupných zdarma:

a) BABYLON

   Modulární hybridní prostřední pro vývoj expertních systémů. Umožňuje reprezentací znalostí pomocí rámců, pravidel a logiky. Vyžaduje Common Lisp. Pracuje na platformách Mac a UNIX.

b) ES

   Nástroj s podporou dopředného i zpětného řetězení a fuzzy množin. Pracuje na platformě PC.

c) GEST (Generic Expert System Tool)

   Podporuje dopředné i zpětné řetězení. Zahrnuje rámce, pravidla a procedury, také fuzzy logiku a faktory určitosti.

d) CLIPS (C Language Integrated Production System)

   viz dále

e) NACLIPS (DYNAamic CLIPS Utilities)

   Doplňky CLIPSu

f) FuzzyCLIPS

   Verze CLIPSu podporující práci s neurčitostí.

g) RT-Expert for DOS, Personal Edition

   Pravidlový systém umožňující integraci expertního systému s kódem psaným v C či C++.

Z komerčních prostředí stojí za zmínku Analyser, EXSYS Professional, KEE, M.4, OPS83, RT-Expert či XpertRule.

3.13 Neurčitost v expertních systémech

Neurčitost může být definována jako nedostatek informací potřebných pro určité rozhodnutí. Jedná se o důležitý problém, neboť díky němu může dojít k chybnému rozhodnutí. V expertních systémech se neurčité informace mohou vyskytovat jak v bázi dat, tak i ve znalostní bázi.

Zdroji neurčitosti jsou nepřesnost, nekonzistence dat, nekompletnost, vágní pojmy, nejisté znalosti, nejednoznačnost, chyby v měřené či chybné usuzování. Reprezentována a zpracovávána bývá pomocí různých aparátů – klasickou pravděpodobností, Bayesovským teorémem, Dempsterovou-Shaferovou teorií, fuzzy logikou či faktory určitosti. Každá z těchto metod má své výhody i nevýhody.

V expertních systémech se objevují dvě možnosti reprezentace neurčitosti, a to pomocí jednoho nebo dvou čísel, která určují, do jaké míry jsme si jisti platností hypotézy. Existují také systémy, kde je neurčitost vyjádřena kvalitativně.

a) Klasická pravděpodobnost

Je nejstarším nástrojem pro zpracování neurčitosti a pochází již ze 17. století. Míra pravděpodobnosti P udává, s jakou jistotou nastane určitý jev E. Platí 3 základní axiomy:

1. pravděpodobnost jevu E nabývá hodnot z intervalu <0,1>,
2. součet pravděpodobností všech závislých jevů je roven 1,
3. pravděpodobnost dvou nezávislých jevů E1 a E2 je rovna součtu jejich pravděpodobností.

Chceme-li definovat pravděpodobnosti v případě vzájemně se ovlivňujících jevů, hovoříme o podmíněné pravděpodobnost, tedy o pravděpodobnosti jevu A za podmínky, že nastane jev B. Pro podmíněnou pravděpodobnost se v expertních systémech využívá Bayesova teorému a z něj odvozených pravidel.

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

Po doménovém expertovi se vyžaduje, aby vložil do systému základní hodnoty (konkrétně P(H/E), P(H/~E), P(H) a P(E)). S pomocí těchto hodnot lze sestavit rozhodovací strom expertního systému. Tento přístup byl použit například v expertním sytému PROSPECTOR.

Mezi velké nevýhody bayesovského přístupu je fakt, že nedostatečně zpracovává neexistující znalost. V případě neznalosti přiděluje dle principu indiference určitou pravděpodobnost, například u 2 hypotéz přiřadí každé z nich hodnotu P=0,5. V reálném světě je ale 50% pravděpodobnost poměrně velká a uživatele takové rozhodnutí expertního systému, ačkoliv založené na nulové znalosti, může vést například k velkým investicím. Rovněž tak je v reálném světě sporné pravidlo, že pravděpodobnost případu a jeho negace se vždy doplňují, tedy že platí P(E) = 1-P(~E).

b) Faktory určitosti

V této metodě zpracování určitosti je s každým pravidlem E spojen faktor určitosti. Tento faktor je určen pomocí míry důvěry MB a míry nedůvěry MD a vyjadřuje stupeň důvěry v hypotézu H, je-li předpoklad E pravdivý.

Faktor určitosti CF může nabývat hodnot z intervalu <-1,1>. CF=1 znamená absolutní důvěru, CF=0 znamená absolutní nedůvěru.

Míry MB a MD mohou nabývat hodnot z intervalu <0,1> a vyjadřují, nakolik je důvěra (resp. nedůvěra) v hypotézu H podporována předpokladem E. Hodnota CF se stanoví podle pravidla CF = MB – MD.

Pravdivá hypotéza má hodnoty MB = 1, MD = 0, a tedy CF = 1.

Nepravdivá hypotéza má hodnoty MB = 0, MD = 1, a tedy CF = -1.

Neexistence znalosti je v tomto případě reprezentována hodnotami MB = 0, MD = 0 a CF = 0.

Metoda faktorů určitosti byla použita například v expertním systému MYCIN.

c) Dampsterova-Shaferova teorie

Tato teorie je založena na pojmu prostředí, který je analogický pojmu univerzum z teorie množin. Prostředí je tvořeno nezávislými (disjunktními) elementy – hypotézami. Každé hypotéze je přiřazeno množství m, neboli míra důvěry, že tato hypotéza platí. Tato hodnota nijak nevypovídá o míře důvěry v ostatní hypotézy daného prostředí.

Množství prázdné podmnožiny m(∅) je obvykle rovno 0, hodnota m nabývá hodnot z intervalu <0,1>.

Při existenci více případů dochází ke zpřesňování věrohodnosti pomocí kombinování množství. K tomu se používá Dempsterovo kombinační pravidlo. Kombinovat lze například množství zjištěná od různých expertů, z opakovaných měření apod..

U této teorie je neurčitost vyjádřena dvěma čísly – mírou domnění a mírou věrohodnosti. Míra domnění v platnost hypotézy H představuje součet množství všech podmnožin množiny H. Míra věrohodnosti potom představuje míru chyby při zamítnutí hypotézy H.

d) Fuzzy přístupy

Fuzzy teorie vychází ze skutečnosti, že v reálném světě většinou neexistují jasné hranice pro zařazování prvků do určitých skupin. Ne vždy určitý prvek jasně

přísluší do určité množiny, častěji může říci, že do některých přísluší více, do jiných méně. Typicky se tak děje pomocí slovních výrazů jako skoro, částečně, převážně či spíše. Fuzzy přístupy, které jsou založeny na teorii fuzzy množin, vícehodnotové logice a lingvistických proměnných, umožňují tyto mlhavé informace zprostředkovat pro účely výpočetní techniky.

Fuzzy množina je charakterizována funkcí příslušnosti , která přiřazuje každému prvku reálné číslo z intervalu <0,1>. Tato hodnota představuje stupeň příslušnosti prvku k množině. Stupeň příslušnosti k množině se nejčastěji vyjadřuje funkcí příslušnosti. Tato funkce má charakteristické tvary, podle nichž jsou jednotlivé typy nazvány. Jedná se nejčastěji o tyto funkce příslušnosti: L-funkce, Λ-funkce, Π-funkce, Γ-funkce, S-funkce a Z-funkce.

Například chceme-li určit stupeň příslušnosti k množině označované vágním pojem „mladý člověk“ , lze stupeň příslušnosti pro jednotlivé hodnoty věku vyjádřit následovně: 20 let – 1,0; 25 let – 1,0; 30 let – 0,8; 35 let – 0,6 let; 40 let – 0,4; 45 let – 0,2 a 50 let 0,0. Po zakreslení do grafu by bylo zřejmé, že v tomto případě se jedná o L-funkci příslušnosti.

Výše uvedený proces převodu dat zatížených neurčitostí na fuzzy množiny charakterizované funkcemi příslušnosti se nazývá fuzzyfikace. Tento proces, nutný pro inferenci s fuzzy hodnotami, je součástí všech fuzzy expertních systémů.

Nad fuzzy množinami lze provádět jak operace analogické s operacemi nad klasickými množinami – doplněk, průnik a sjednocení (NOT, AND, OR), tak také operace charakteristické právě pro fuzzy množiny.

Po skončení procesu iterace je potřeba provést defuzzyfikaci, tedy převést fuzzy výsledky na konkrétní číselné hodnoty.

Typickým expertním systémem, umožňující práci s fuzzy množinami, je FuzzyCLIPS, vyvinutý v National Research Council Canada, který lze volně získat na internetu, a to v současné době ve verzích z roku 2004 - 6.10d pro platformu Windows a 6.10c pro platformy UNIX, Solaris a VAX. V současnosti je tento projekt pozastaven.

4 Prostředí CLIPS

4.1 Historie CLIPS

Myšlenka na vytvoření CLIPSu (Language Integrated Production System) se datuje do roku 1984 a jeho autory jsou vývojáři Johnsonova vesmírného střediska NASA. Důvodem jeho vzniku bylo, že v té době téměř všechny expertní systémy využívaly jazyk LISP, který byl pro účely NASA shledán nevyhovujícím. Důvodem byla nedostupnost LISPu pro různorodé konvenční počítače, vysoká cena nejmodernějších nástrojů LISPu a hardware, a také nedostatečná integrace LISPu s ostatními jazyky.

Vývojáři z oddělení umělé inteligence NASA cítili, že použití konvenčního jazyka, jakým je C, by eliminovalo většinu z těchto nevýhod. Ačkoliv na trhu existovaly nástroje na bázi jazyka C, jejich vysoká cena a dlouhá lhůta dodání vedla NASA k rozhodnutí vytvořit si vlastní expertní systém.

Po dvou měsících, v roce 1985, byl vyvinut první prototyp. Primární pozornost byla věnována kompatibilitě s v té době vyvíjenými expertními systémy NASA. Proto byla syntaxe CLIPSu podobná syntaxi systému ART od společnosti Inference Corporation. Ačkoliv měl prototyp dost chyb, dokázal, že myšlenka celého projektu je správná a má velkou šanci na úspěch. Rok dalšího vývoje zlepšil přenositelnost, výkon a funkčnost CLIPSu. V létě 1986 byl systém CLIPS, verze 3.0, vypuštěn i mimo NASA.

Verze 4.0 a 4.1, které byly představeny v roce 1987, poskytovaly výrazně vyšší výkon, integraci externích jazyků a rychlejší možnosti dodávky.

Verze 5.0 z jara 1991 zahrnovala dvě nová programovací paradigmata – procedurální programování a objektově orientované programování. Objektově orientovaný jazyk poskytovaný CLIPSem se nazývá COOL (CLIPS Object-Oriented Language).

Verze 5.1 umožnila podporu některých nových rozhraní, verze 6.0 z roku 1993 zahrnovala podporu modulárních programů a úzké propojení mezi objektově orientovanými a pravidlovými systémy CLIPSu. Verze 6.1 z roku 1998 odstranila podporu některých zastaralých překladačů jazyků, naopak přibyla podpora překladačů jazyka C++.

Ve verzi 6.2 z roku 2002 byla přidána podpora dalších prostředí a došlo k vylepšení vývojových rozhraní pro Windows XP a MacOS.

V současné době dostupná verze 6.3 BETA nabízí vylepšený výkon, rozšířenou jazykovou podporu a datový typ integer o délce 64-bitů.

Díky svým vlastnostem nalezl CLIPS široké uplatnění v mnoha oblastech, především ve vládních, průmyslových a akademických institucích [16].

4.2 Charakteristika CLIPSu

CLIPS je nástrojem pro vývoj expertních systémů. Umožňuje vytvářet systémy jak pravidlové, tak objektově orientované. Díky skutečnosti, že je napsán v jazyce C, je zajištěna jeho snadná přenositelnost mezi platformami a dobrá kompatibilita s ostatními systémy. Může být doplněn procedurálním kódem (tzv. subprocedurou) a umožňuje rozšíření uživatelem prostřednictvím různých protokolů. Poskytuje jak textově, tak graficky založené uživatelské rozhraní, dále ladící nástroje, on-line nápovědu, integrované editory a vysvětlovací mechanismy. K dispozici je řada dokumentačního materiálu, jako uživatelská příručka či referenční manuál. Jedná se o zdarma dostupný software stáhnutelný z internetu, případně lze získat za mírný poplatek verzi na CD.

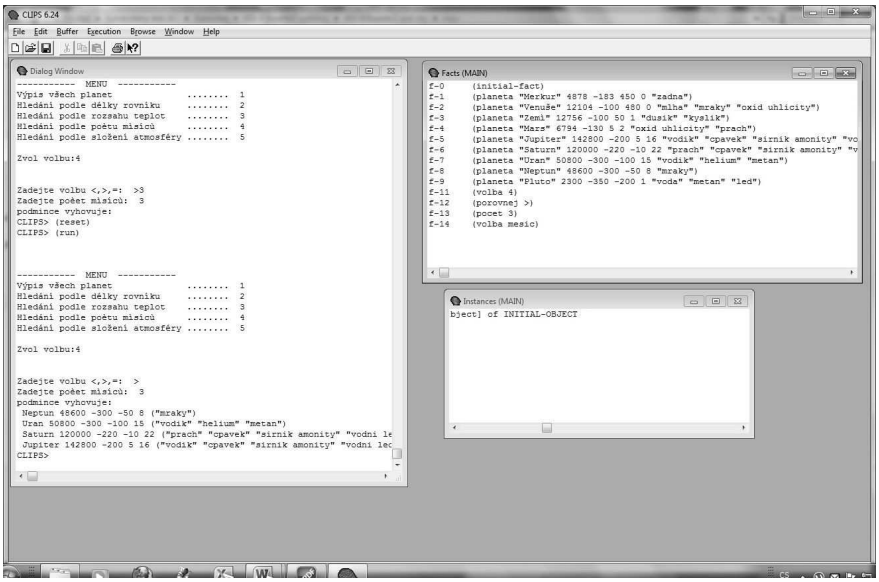
CLIPS umožňuje 3 možnost pro reprezentaci znalostí:

1) PRAVIDLA – jsou primárně určená pro reprezentaci heuristických znalostí získaných zkušeností

2) DEFINIČNÍ A GENERICKÉ FUNKCE – jsou určeny pro reprezentaci procedurálních znalostí

3) OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ – je rovněž určeno pro reprezentaci procedurálních znalostí. Základem jsou tyto vlastnosti objektově orientovaného programování – třídy, výměna zpráv, abstrakce, zapouzdření, dědičnost a polymorfismus.

Expertní systém může být vyvíjen buď jako pravidlový, nebo jako objektově orientovaný, případně jako kombinace obou těchto přístupů.



Obr. 4 – Prostředí CLIPS

CLIPS může být rovněž volán procedurálním programem, vykoná požadovanou funkci a opět vrátí řízení volajícímu programu. Stejně tak může být z CLIPSu volán externí procedurální program. Poté, co externí program vykoná požadovanou funkci, vrací se řízení opět CLIPSu.

4.3 Další varianty prostředí CLIPS

Kromě zmíněného základního prostředí CLIPS existují ještě další verze, z nichž stojí za zmínku především tyto dvě:

a) WxClips

Toto rozšíření CLIPSu vyvinul v roce 1992 Julian Smart z Artificial Intelligence Applications Institute, který je součástí Edinburghské univerzity. Jedná se o volně dostupné grafické prostředí, umožňující spouštění kódů CLIPSu. Klasický CLIPS rozšiřuje o stovky funkcí, umožňujících tvorbu grafického rozhraní. K dispozici je rovněž fuzzy verze tohoto prostředí [19].

b) FuzzyCLIPS

Toto již dříve zmíněné prostředí umožňuje CLIPSu zpracovávat neurčitost. Vyvinuto bylo v National Research Council Canada. Následující tabulky představují srovnání mezi klasickým CLIPSem a FuzzyCLIPSem:

SHODY CLIPS a FUZZY CLIPS

CLIPS	FUZZY CLIPS
• také on může pracovat s "fuzzy" fakty, ale implementace je poněkud rozsáhlejší a pracnější viz. ukázkový program **old_clips_money.clp**	• Může operovat se všemi konstrukcemi jako klasický Clips • Možné porovnávání se vzorem • Fakt s položkou fuzzy i nefuzzy

VÝHODY

CLIPS	FUZZY CLIPS
• Možná tvorba expertních systémů • Free • Snadnější orientace v tom, co bude výstupem programu	• Možná tvorba expertních fuzzy systémů • Free • Zachycuje mlhavost a nejistotu našeho světa jednodušeji než klasický Clips • Jednoduše operuje s nejistotou (= CF faktor jistoty) • Využívá ty samé struktury jako klasický Clips • Možnost vykreslení tvaru fuzzy množiny programátorovi

NEVÝHODY

CLIPS	FUZZY CLIPS
• Kolize pravidel • Je možné zachycovat fuzzy pojmy, ale komplikovaně (a navíc bez matematického základu) • Lze vyjádřit faktor nejistoty (CF), ale jako definici dalšího faktu • Neužívá českou diakritiku • Case-sensitive • rozsáhlost programu => obtížněji se v něm vyznáme • jen textový režim	• Neužívá českou diakritiku • Case-sensitive • Obtížněji se odhaduje, co bude výsledkem – výstupem programu • Rozsáhlost programu => obtížněji se v něm vyznáme • Jen textový režim

Tab. 2 – Srovnání CLIPS a FuzzyCLIPS [2]

5 Expertní systém pro výpočet opravných položek k pohledávkám

5.1 Popis řešeného problému

Problematika tvorby opravných položek k pohledávkám spadá do oblastí účetnictví a nejčastěji se s ní lze setkat při zpracovávání účetních uzávěrek. Ačkoliv jsou pravidla pro tvorbu opravných položek k pohledávkám jasně definována v Zákoně č. 593/1992 Sb. o rezervách pro zjištění základu daně z příjmů a v Českém účetním standardu pro podnikatele č. 005 (případně dalších standardech pro nepodnikatelské subjekty), není v praxi vždy jednoznačně chápána a často vyžaduje delší praxi a zkušenosti v oboru. Vzhledem k tomu, že pravidla pro tvorbu opravných položek lze snadno převést do pravidel ve formátu IF-THEN, je právě tato oblast vhodná pro nasazení jednoduchého expertního systému vytvořeného v prostředí CLIPS.

5.2 Problematika opravných položek k pohledávkám

Opravné položky k pohledávkám slouží pro dočasné snížení hodnoty pohledávky vykazované v účetnictví. Protože v účetnictví musí být dodržována zásada opatrnosti, je nutné v účetních výkazech vykazovat pohledávky v takové výši, která odpovídá realitě. Tato se velmi často liší od původní výše pohledávky. Pokud například účetní jednotka ví, že je pohledávka již půl roku po lhůtě splatnosti, a že pravděpodobnost, že bude uhrazena, je rovna 20%, je nutné tuto skutečnost vyjádřit právě pomocí opravné položky k pohledávce.

Existují dva druhy opravných položek k pohledávkám -daňové opravné položky, které jsou nákladem, o který si lze snížit základ daně z příjmů a účetní opravné položky, které daňově uznatelným nákladem nejsou. Postup účetních jednotek je zpravidla takový, že nejprve zjistí, do jaké míry je možné k pohledávce vytvořit daňovou opravnou položku (která je pro ně výhodnější), a následně dotvoří účetní opravnou položku do takové výše, aby celková výše opravné položky odpovídala skutečné hodnotě pohledávky k danému okamžiku.

Výše daňových opravných položek je dána zákonem a může se pohybovat od 0% do 100% hodnoty pohledávky. Naopak výši účetních opravných položek si stanovuje každá účetní jednotka sama, a to nejčastěji ve vnitropodnikových směrnicích.

5.3 Popis expertního systému

Následující expertní systém, jehož kompletní verze je součástí přílohy, má za cíl na základě vstupních dat z účetního systému doporučit uživateli vhodnou výši daňových a účetních opravných položek.

Vstupem jsou uživatelem zadané informace o konkrétní pohledávce, a to:

a) výše pohledávky

b) doba, která uplynula od splatnosti pohledávky (reprezentována časovými intervaly)

c) informace, zda byla v loňském roce vytvořena opravná položka dle §8c (Lze říci, že se jedná o nejvýhodnější možnost, protože tento paragraf umožňuje za splnění určitých podmínek tvořit daňově uznatelnou opravnou položku ve výši 100%. Pokud byla podle tohoto paragrafu vytvořena opravná položka v loňském roce, zůstává tato v platnosti i v letech následujících.)

d) informace, zda lze tvořit daňovou opravnou položku (Zákon jasně stanoví, ke kterým pohledávkám daňově uznatelnou opravnou položku tvořit nelze – např. k pohledávkám vůči spojeným osobám apod.)

e) informaci, zda je ve věci pohledávky vedeno soudní řízení (tímto je myšleno rozhodčí řízení, soudní řízení, nebo správní řízení, kterého se věřitel řádně účastní).

f) informaci, zda není pohledávka řádně přihlášena do insolvenčního řízení.

V CLIPSu je vstup dat řešen následovně:

```
(defrule zadej_udaje

=>

(printout t "Zadejte vysi pohledavky - oddelovac desetinnych
mist
 je tecka:" crlf)
(bind ?castka (read))

(printout t "Zadejte pocet mesicu po splatnosti nasledovne:"
crlf)
(printout t " 0 -  3 mesicu ..... zadejte 0" crlf)
(printout t " 3 -  6 mesicu ..... zadejte 3" crlf)
(printout t " 6 - 12 mesicu ..... zadejte 6" crlf)
(printout t "12 - 18 mesicu ..... zadejte 12" crlf)
(printout t "18 - 24 mesicu ..... zadejte 18" crlf)
(printout t "24 - 30 mesicu ..... zadejte 24" crlf)
(printout t "30 - 36 mesicu ..... zadejte 30" crlf)
(printout t "nad  36 mesicu ..... zadejte 36" crlf)
(bind ?inter (read))
(while (not (or (eq ?inter 0) (eq ?inter 3) (eq ?inter 6)
 (eq ?inter 12) (eq ?inter 18) (eq ?inter 24) (eq ?inter 30)
 (eq ?inter 36)))
 (printout t "Mozne hodnoty 0,3,6,12,18,24,30 nebo 36!
Zadejte
  znovu:" crlf)
 (bind ?inter (read)))

(printout t "Byl minuly rok pouzit §8c? (a/n):" crlf)
(bind ?mr8c (read))
(while (not (or (eq ?mr8c a) (eq ?mr8c n)))
 (printout t "Byl minuly rok pouzit §8c? (a/n):" crlf)
 (bind ?mr8c (read)))

(printout t "Lze tvorit danovou opravnou polozku? (a/n):"
crlf)
(bind ?doplze (read))
(while (not (or (eq ?doplze a) (eq ?doplze n)))
 (printout t "Lze tvorit danovou opravnou polozku? (a/n):"
crlf)
 (bind ?doplze (read)))

(printout t "Je vedeno soudni ci insolvencni rizeni? (a/n):"
crlf)
(bind ?soud (read))
(while (not (or (eq ?soud a) (eq ?soud n)))
 (printout t "Je vedeno soudni ci insolvencni rizeni? (a/n):"
```

```
  crlf)
 (bind ?soud (read)))

(printout t "Je pohledavka radne prihlasena v insolvencnim
rizeni? (a/n):" crlf)
(bind ?insol (read))
(while (not (or (eq ?insol a) (eq ?insol n)))
 (printout t "Je pohledavka radne prihlasena v insolvencnim
  rizeni? (a/n):" crlf)
 (bind ?insol (read)))

(assert (castka ?castka))
(assert (inter ?inter))
(assert (mr8c ?mr8c))
(assert (doplze ?doplze))
(assert (soud ?soud))
(assert (insol ?insol))
(assert (pomocuop 0)))
```

Výstupem expertního systému je informace o výši daňové a účetní opravné položky v procentech. V případě, že lze vytvořit opravnou položku podle §8c expertní systém na tuto možnost upozorní. V tomto případě je totiž na samotném uživateli, aby si sám vybral, ke kterým konkrétním pohledávkám od stejného dlužníka bude v daném roce opravnou položku podle §8c tvořit.

Výstupy jsou v syntaxi CLIPSu řešeny takto:

```
(defrule vypis1

 (dopar ?dopar)
 (test (neq ?dopar 0))
 (paraar ?paraar)
 (uopar ?uopar)
=>
(printout t "Danova OP ve vysi " ?dopar "% vytvorena podle §"
 ?paraar ". Ucetni OP je ve vysi " ?uopar "%." crlf))


(defrule vypis2

 (doporuc8c a)
=>
(printout t "Lze doporucit 100% danovou OP dle §8c do vyse
max.
 30000 Kc na dluznika." crlf))
```

```
(defrule vypis3

 (dopar 0)
 (dopar ?dopar)
 (uopar ?uopar)
=>
(printout t "Danova OP nemohla byt vytvorena. Ucetni OP je ve
vysi
 " ?uopar "%." crlf))
```

Pro praktické využití tohoto systému v prostředí větších účetních jednotek by bylo potřebné provést nenáročnou úpravu kódu, kdy by vstup byl načítán z externího vstupního souboru, pocházejícího z účetního systému, a výsledné fakty by byly ukládány rovněž do externího souboru, určeného k importu do účetního systému.

5.4 Daňově uznatelné opravné položky

V Zákoně č. 593/1992 Sb. o rezervách pro zjištění základu daně z příjmů jsou pravidla pro tvorbu daňově uznatelných opravných položek definována takto [18]:

§ 8 Opravné položky k pohledávkám za dlužníky v insolvenčním řízení

(1) Opravné položky k pohledávkám za dlužníky v insolvenčním řízení, které jsou výdajem (nákladem) na dosažení, zajištění a udržení příjmů), mohou vytvořit poplatníci daně z příjmů, kteří vedou účetnictví, až do výše rozvahové hodnoty nepromlčených pohledávek přihlášených u soudu od zahájení insolvenčního řízení do konce lhůty stanovené v rozhodnutí soudu o úpadku nebo do konce lhůty podle insolvenčního zákona, spojí-li soud s rozhodnutím o úpadku rozhodnutí o povolení oddlužení, a to v období, za které se podává daňové přiznání a v němž byly přihlášeny. Byla-li povolena reorganizace, namísto přihlášky pohledávky postačí, že dlužník věřitelovu pohledávku správně uvedl v seznamu svých závazků podle zvláštního právního předpisu. K pohledávkám vyloučeným v § 2 odst. 2 nelze tvořit opravné položky, které jsou výdajem (nákladem) na dosažení, zajištění a udržení příjmů podle tohoto ustanovení.

(2) Opravné položky se zruší v návaznosti na výsledky insolvenčního řízení nebo v případě, že pohledávku účinně popřel insolvenční správce, věřitel nebo dlužník a zvláštní právní předpis těmto osobám právo popřít pohledávku přiznává.

(3) Jestliže pominou důvody pro existenci opravné položky vytvořené podle tohoto ustanovení nebo na základě rozhodnutí poplatníka, je možné snížit vytvořenou opravnou položku na úroveň, která by mohla být vytvořena podle ustanovení § 8a. Poplatník pak pokračuje v tvorbě opravné položky podle § 8a.

§ 8a Opravné položky k nepromlčeným pohledávkám splatným po 31. prosinci 1994

(1) Opravné položky k nepromlčeným pohledávkám splatným po 31. prosinci 1994, jejichž rozvahová hodnota v okamžiku vzniku nepřesáhne částku 200 000 Kč a jejichž tvorba je výdajem (nákladem) na dosažení, zajištění a udržení příjmů mohou v období, za které se podává daňové přiznání, vytvářet poplatníci daně z příjmů, kteří vedou účetnictví, pokud k těmto pohledávkám nevytvářejí opravné položky a rezervy podle § 5 a 5a, a od konce sjednané lhůty splatnosti pohledávky uplynulo více než 6 měsíců, až do výše 20 % neuhrazené rozvahové hodnoty pohledávky.

(2) Vyšší opravné položky, než je uvedeno v odstavci 1, lze vytvářet ke zde uvedeným pohledávkám jen v případě, bylo-li ohledně těchto pohledávek zahájeno rozhodčí řízení podle zvláštního právního předpisu nebo soudní řízení a nebo správní řízení podle zvláštního právního předpisu, jehož se poplatník daně z příjmů řádně účastní za podmínky, že od konce sjednané lhůty splatnosti pohledávky uplynulo více než

a) 12 měsíců, až do výše 33 % neuhrazené rozvahové hodnoty pohledávky,
b) 18 měsíců, až do výše 50 % neuhrazené rozvahové hodnoty pohledávky,
c) 24 měsíců, až do výše 66 % neuhrazené rozvahové hodnoty pohledávky,
d) 30 měsíců, až do výše 80 % neuhrazené rozvahové hodnoty pohledávky,
e) 36 měsíců, až do výše 100 % neuhrazené rozvahové hodnoty pohledávky.

(3) Opravné položky k nepromlčeným pohledávkám splatným po 31. prosinci 1994, jejichž rozvahová hodnota v okamžiku vzniku je vyšší než 200 000 Kč a nejsou k nim vytvářeny opravné položky podle § 5 a 5a, mohou v období, za které se podává daňové přiznání, vytvářet poplatníci daně z příjmů, kteří vedou účetnictví, jen v případě, bylo-li ohledně těchto pohledávek zahájeno rozhodčí řízení podle zvláštního právního předpisu nebo soudní řízení a nebo správní řízení podle zvláštního právního předpisu, jehož se poplatník daně z příjmů řádně účastní za podmínky, že od konce sjednané lhůty splatnosti pohledávky uplynulo více než

a) 6 měsíců, až do výše 20 % neuhrazené rozvahové hodnoty pohledávky,
b) 12 měsíců, až do výše 33 % neuhrazené rozvahové hodnoty pohledávky,
c) 18 měsíců, až do výše 50 % neuhrazené rozvahové hodnoty pohledávky,
d) 24 měsíců, až do výše 66 % neuhrazené rozvahové hodnoty pohledávky,
e) 30 měsíců, až do výše 80 % neuhrazené rozvahové hodnoty pohledávky,
f) 36 měsíců, až do výše 100 % neuhrazené rozvahové hodnoty pohledávky.

(4) Opravné položky podle odstavců 1 až 3 nelze uplatnit u pohledávek již odepsaných na vrub výsledku hospodaření a dále u pohledávek vzniklých

a) za společníky, akcionáři, členy družstev za upsaný vlastní kapitál,
b) mezi spojenými osobami vymezenými v zákoně o daních z příjmů.

(5) Opravné položky vytvořené podle odstavců 1 až 3 se zruší, pokud pominou důvody pro jejich existenci nebo pokud pohledávka, k níž byla opravná položka vytvořena, se promlčela, popřípadě nastaly důvody, za nichž se odpis pohledávky považuje za výdaj (náklad) na dosažení, zajištění a udržení příjmů podle ustanovení zákona o daních z příjmů.

§ 8b Opravné položky k pohledávkám z titulu ručení za celní dluh

(1) Opravné položky podle § 8 a 8a tohoto zákona mohou vytvářet poplatníci daně z příjmů, kteří vedou účetnictví a kteří podle celního zákona ručí za celní dluh,

k pohledávkám vzniklým z titulu ručení za celní dluh (tj. zajištění celního dluhu) podle celního zákona.

(2) Opravné položky podle odstavce 1 lze vytvářet jen do výše hodnoty pohledávky odpovídající provedené úhradě celního dluhu.

(3) Opravné položky ručitel nemůže vytvářet ze splněné pohledávky z titulu ručení, pokud dluh za dlužníka nesplní v době splatnosti určené celními orgány

§ 8c

Nepostupuje-li poplatník u nepromlčené pohledávky podle § 5, 5a, 6, 8, 8a a 8b, může v období, za které se podává daňové přiznání, vytvořit opravnou položku až do výše 100 % její neuhrazené rozvahové hodnoty bez příslušenství pouze v případě, že
a) se nejedná o pohledávku vymezenou v § 8a odst. 4,
b) rozvahová hodnota pohledávky bez příslušenství v okamžiku jejího vzniku nepřesáhne částku 30 000 Kč.
c) od konce sjednané lhůty splatnosti pohledávky uplynulo nejméně 12 měsíců, a
d) celková hodnota pohledávek bez příslušenství vzniklých vůči témuž dlužníkovi, u nichž uplatňuje postup podle tohoto ustanovení, nepřesáhne za období, za které se podává daňové přiznání, částku 30 000 Kč.

O pohledávce, k níž byla vytvořena opravná položka podle tohoto ustanovení, je poplatník povinen vést samostatnou evidenci.

Ukázka z řešení pro §8:

```
(defrule insolvence
 (declare (salience 3))
 (insol a)
=>
(assert (paraar 8))
(assert (dopar 100)))
```

Ukázka z řešení pro §8a odst. 1:

```
(defrule pravidlo8
 (danove_bez_soudu a)
 (castka ?x)
 (test (<= ?x 200000))
 (mr8c n)
 (insol n)
=>
(assert (paraar 8a1)))
```

Ukázka z řešení pro §8c:

```
(defrule paragraf8cnove

 (declare (salience 1))
 (castka ? castka)
 (test (<= ?castka 30000))
 (not (inter 0|3|6))
 (not (mr8c a))
 (not (insol a))
=>
(assert (doporuc8c a)))

(defrule paragraf8c

 (paraar 8c)
=>
(assert (dopar 100)))
```

5.5 Účetní opravné položky

Pro účely níže popsaného expertního systému jsou stanovena pravidla pro tvorbu účetních opravných položek následovně:

od splatnosti uběhlo	výše celkové OP
0 – 3 měsíce	0%
3 – 6 měsíců	25%
6 – 12 měsíců	50%
více než 12 měsíců	100%

kde výše celkové opravné položky je součtem daňové a účetní opravné položky.

Ukázka pravidel pro dopočet účetní opravné položky v CLIPSu:

```
(defrule pravidlouop1

 (dopar 20)
 (inter ?inter)
=>
(if (= ?inter 6)
  then (assert (uopar 30))
  else (assert (uopar 80))))

(defrule pravidlouop2

 (dopar ?x)
 (pomocuop ?y)
 (test (or (eq ?x 33) (eq ?x 50) (eq ?x 66) (eq ?x 80)
  (eq ?x 100)))
=>
(assert (uopar (- 100 ?x))))
```

5.6 Ukázky výstupů expertního systému

Příklad 1

```
CLIPS> (reset)
CLIPS> (run)
Zadejte vysi pohledavky - oddelovac desetinnych mist je tecka:
100000
Zadejte pocet mesicu po splatnosti nasledovne:
 0 -  3 mesicu ..... zadejte 0
 3 -  6 mesicu ..... zadejte 3
 6 - 12 mesicu ..... zadejte 6
12 - 18 mesicu ..... zadejte 12
18 - 24 mesicu ..... zadejte 18
24 - 30 mesicu ..... zadejte 24
30 - 36 mesicu ..... zadejte 30
nad  36 mesicu ..... zadejte 36
2
Mozne hodnoty 0,3,6,12,18,24,30 nebo 36! Zadejte znovu:
18
Byl minuly rok pouzit §8c? (a/n):
n
Lze tvorit danovou opravnou polozku? (a/n):
a
Je vedeno soudni, rozhodci ci správni rizeni, jehoz se radne ucastnite? (a/n):
a
Je pohledavka radne prihlasena v insolvencnim rizeni? (a/n):
n
Danova OP ve vysi 50% vytvorena podle §8a2. Ucetni OP je ve vysi 50%.
CLIPS>
```

Pohledávka v částce 100.000,- Kč, která je 18 měsíců po lhůtě splatnosti (úmyslně pro ukázku byl nejprve zadán neexistující časový interval), lze k ní tvořit daňově uznatelnou opravnou položku a je vedeno soudní řízení (byl vydán platební rozkaz). Systém doporučuje vytvořit daňově uznatelnou opravnou položku podle § 8a odst. 2 ve výši 50% a dále účetní opravnou položku do celkové výše 100%, tedy ve výši 50%.

Příklad 2

```
CLIPS> (reset)
CLIPS> (run)
Zadejte vysi pohledavky - oddelovac desetinnych mist je tecka:
5000
Zadejte pocet mesicu po splatnosti nasledovne:
 0 -  3 mesicu ..... zadejte 0
 3 -  6 mesicu ..... zadejte 3
 6 - 12 mesicu ..... zadejte 6
12 - 18 mesicu ..... zadejte 12
18 - 24 mesicu ..... zadejte 18
24 - 30 mesicu ..... zadejte 24
30 - 36 mesicu ..... zadejte 30
nad  36 mesicu ..... zadejte 36
30
Byl minuly rok pouzit §8c? (a/n):
n
Lze tvorit danovou opravnou polozku? (a/n):
a
Je vedeno soudni, rozhodci ci správni rizeni, jehoz se radne ucastnite? (a/n):
n
Je pohledavka radne prihlasena v insolvencnim rizeni? (a/n):
n
Lze doporucit 100% danovou OP dle §8c do vyse max. 30000 Kc na dluznika.
Danova OP ve vysi 20% vytvorena podle §8a1. Ucetni OP je ve vysi 80%.
CLIPS>
```

Pohledávka v částce 5.000,- Kč, která je 30 měsíců po lhůtě splatnosti, lze k ní tvořit daňově uznatelnou opravnou položku. Soudní řízení vedeno není z důvodu malé částky pohledávky a ani není přihlášena v insolvenčním řízení. Protože tato pohledávka splňuje podmínky pro tvorbu daňově uznatelné opravné položky ve výši 100% podle § 8c, systém přednostně doporučuje právě tuto tvorbu, která je pro společnost výhodnější. Pokud by společnost nevyužila tuto možnost, doporučuje systém tvořit daňově uznatelnou opravnou položku podle § 8a odst. 1 ve výši 20% a dále účetní opravnou položku do celkové výše 100%, tedy ve výši 80%.

Příklad 3

```
CLIPS> (reset)
CLIPS> (run)
Zadejte vysi pohledavky - oddelovac desetinnych mist je tecka:
5000
Zadejte pocet mesicu po splatnosti nasledovne:
 0 -  3 mesicu ..... zadejte 0
 3 -  6 mesicu ..... zadejte 3
 6 - 12 mesicu ..... zadejte 6
12 - 18 mesicu ..... zadejte 12
18 - 24 mesicu ..... zadejte 18
24 - 30 mesicu ..... zadejte 24
30 - 36 mesicu ..... zadejte 30
nad  36 mesicu ..... zadejte 36
12
Byl minuly rok pouzit §8c? (a/n):
a
Lze tvorit danovou opravnou polozku? (a/n):
a
Je vedeno soudni, rozhodci ci správni rizeni, jehoz se radne ucastnite? (a/n):

n
Je pohledavka radne prihlasena v insolvencnim rizeni? (a/n):
n
Danova OP ve vysi 100% vytvorena podle §8c. Ucetni OP je ve vysi 0%.
CLIPS>
```

Pohledávka v částce 5.000,- Kč, která je 12 měsíců po lhůtě splatnosti a lze k ní tvořit daňově uznatelnou opravnou položku. Protože v minulém roce již byla vytvořena daňově uznatelná opravná položka ve výši 100% podle § 8c, systém doporučuje takto vytvořenou opravnou položku ponechat. Účetní opravná položka se již netvoří.

6 Závěr

Expertní systémy představují v mnoha oblastech optimální programový prostředek pro řešení situací, které by jinak vyžadovaly lidské experty. Nasazení těchto systémů zpravidla znamená výrazné zefektivnění procesů a snížení nákladů. Uživatelé těchto řešení mají k dispozici trvale dostupnou, kvalitní a rychlou expertízu, což v případě využití lidských expertů nebývá samozřejmostí. V případě expertního systému je navíc k dispozici také zpětný doklad postupu při hledání řešení. Další výhodu představuje nasazení expertních systémů schopných dalšího učení, které průběžně obohacují svoji znalostní základnu o nové poznatky, čímž zajišťují stálou aktuálnost svých výstupů.

V technologicky a vědecky náročnějších oblastech, jako jsou například výroba, medicína, průmysl, obchod, životní prostředí a energetika, zemědělství, telekomunikace, vojenství, letectví či kosmonautika, je využití expertních systémů poměrně rozšířené. V těchto případech se expertní systémy zpravidla nepoužívají samostatně, ale jako součásti komplexních výpočetních systémů, navíc bývají úzce propojeny s dalšími oblastmi umělé inteligence, jako jsou generické algoritmy či neuronové sítě.

V současné době probíhají po celém světě pravidelné semináře a konference, věnované jak umělé inteligenci obecně, tak i přímo expertním systémům. Příkladem takové akce je již 22. ročník konference DEXA, která se uskuteční na přelomu září a října roku 2011 ve Francii. Tyto akce svědčí o stálé atraktivnosti a dalším rozvoji tohoto odvětví umělé inteligence.

Zhodnotit současný stav využívání expertních systémů není snadné především proto, že informace o nich nebývají běžně dostupné. Společnosti tyto informace často považují za součást svého duševního vlastnictví a obchodního tajemství. Navíc, ve vojenství a dalších strategických oblastech, se jedná o informace vysloveně důvěrné či tajné. Dovoluji si vyslovit předpoklad, že v těchto případech lze při odhadu stavu vývoje

expertních systémů vycházet z faktu, že vývoj technologií v těchto oblastech bývá vždy o několik kroků napřed než v oblastech civilních.

Obecné povědomí nevědecké veřejnosti o těchto systémech bohužel není příliš vysoké, ačkoliv s expertními systémy přicházíme do styku poměrně běžně. Uživatel často o přítomnosti expertního systému ani neví, přestože jej používá. Příkladem takových expertních systémů jsou webové aplikace pro podporu uživatelů, typicky například Office Assistant od společnosti Microsoft, PerfectExpert od společnosti Corel nebo elektronický nástroj pro zadávání a evidenci veřejných zakázek E-ZAK.

Pomocí volně dostupných prostředí pro expertní systémy, jako je například popisovaný CLIPS, lze i pro potřeby malé organizace dosáhnout efektivního řešení mnoha úkolů, které se standardně, a lze říci že často zbytečně, řeší klasickými procedurálními metodami. Přitom právě expertní systémy představují řešení pomocí usuzovacího procesu, který mnohem více odpovídá uvažování člověka.

Prostředí CLIPS je prostředek umožňující jednoduché, snadno naučitelné a člověku logicky blízké řešení, díky svým možnostem rozšíření o procedurální kód je ale vhodné i pro řešení náročnějších a složitějších úkolů.

Osobně jsem před vypracováním této diplomové práce měla jen velmi vágní povědomí o expertních systémech. Svojí původní představu, že se jedná o spíše teoretickou a dnes již téměř zapomenutou záležitost, jsem v průběhu zpracování práce přehodnotila. Naopak jsem zjistila, že se jedná o velmi zajímavý informatický obor, který znamená velký technologický, finanční i společenský přínos.

Na samotném prostředí CLIPS mne velmi zaujala jeho uživatelská přívětivost a především jednoduchost samotné syntaxe, kterou se podle mne může naučit i člověk,

který nikdy nepřišel do styku s programováním. V tomto přesvědčení jsem se utvrdila i při psaní praktické části této práce – systému pro výpočet opravných položek k pohledávkám. Tento systém jsem již prakticky využila ve své ekonomické praxi a ráda bych v budoucnu prostředí CLIPS využila i pro další praktické aplikace.

7 Použitá literatura

[1] MINSKY, Marvin. *Semantic Information Processing*, MIT Press, 1968, ISBN 0262130440

[2] RICH, Elaine, KNIGHT, Kevin. *Artificial Intelligence*. Columbus : McGraw-Hill College, 1991. 621 s. ISBN 9780070522633.

[3] ČSN ISO/IEC 2382-28 (369001). Informační technologie - Slovník - Část 28: Umělá inteligence - Základní pojmy a expertní systémy, Praha: Český normalizační institut, 1999, 44 s..

[4] TURING, Alan M. Computing Machinery and Intelligence. *Mind*, 1950, vol. 59., s. 433-460.

[5] SEARLE, John R. Minds, Brains and Programs. *The Behavioral and Brain Sciences, 1980, vol.*3, s. 417–424.

[6] SIMON, Herbert, NEWELL, Allen. Heuristic Problem Solving: The Next Advance in Operations Research. *Operations Research, 1*958, vol. 6, s. 1-10.

[7] BERKA, Petr. *Úvod do umělé inteligence – T1: umělá inteligence jako vědní disciplína*. [online]. 2007, str. 15, dostupné na <http://sorry.vse.cz/~berka/docs/4iz229/s01-uvodAI-4p.pdf>.

[8] VONDRÁK, Ivo. *Umělá inteligence a neuronové sítě*. Ostrava : VŠB-TU, Fakulta elektrotechniky a informatiky, 1994.

[9] MIKULECKÝ, Peter; VLČEK, Pavel. *Umělá inteligence a expertní systémy* [online]. VŠB, 2007, s. 40. Dostupné na <lide.uhk.cz/fim/ucitel/mikulpe1/ES/ES1.ppt>.

[10] STREIT, Ulrich. *Introduction to Geoinformatics*. Institute for Geoinformatics of the University of Muenster University of Muenster, 2000.

[11] BUCHANAN, B.G., SMITH, R.G.. *Fundamentals of Expert Systems.* [online]. Annual Review of Computer Science, 1998, Vol. 3: s. 23-58. 1988. Dostupné na <http://media.wiley.com/product_data/excerpt/18/04712933/0471293318.pdf>.

[12] BERKA, Petr. *Tvorba znalostních systémů*. 1. vyd., VŠE Praha, 1994. 190 s..

[13] GIARRATANO, Joseph C. , RILEY, Gary D. . *Expert Systems: Principles and Programming, Fourth Edition*. [s.l.] : Course Technology, 2004. 842 s. ISBN 978-0534384470.

[14] Celbová, Iva. Úvod do problematiky expertních systémů. *Ikaros* [online]. 1999, roč. 3, č. 8 [cit. 22.03.2011]. Dostupný na <http://www.ikaros.cz/node/393>.

[15] ŠTÝBNAROVÁ, Libuše. *Expertní systémy*, Příručka k přednáškám [online]. Opava, UI FPF SU, [cit. 2011-01-10]. Dostupné na <http://ui.fpf.slu.cz/diplomky/znalostni_a_expertni_systemy>.

[16] CULBERT, Chris, RILEY Gary; DONNELL, Brian. *CLIPS Reference Manual, Volume III, Iterfaces Guide, Version 6.24*, 2006.

[17] SILER, William, BUCKLEY, James J.. *Fuzzy Expert Systems and Fuzzy Reasoning*, Wiley-Interscience; 1. edition, 2004, 424 s., ISBN-13: 978-0471388593.

[18] Zákon č. 593/1992 Sb., o rezervách pro zjištění základu daně z příjmů.

[19] SMART, Julian. *WxCLIPS* [online]. [cit. 2010-12-25]. Dostupné na <http://www.anthemion.co.uk/wxclips/>.

Obrázek [1] SAYGIN, A.P., CICEKLI, I., AKMAN, V. Turing Test: 50 years later. *Minds and Machines*, 2000, University of California. Dostupné na <http://crl.ucsd.edu/~saygin/papers/MMTT.pdf>.

Obrázek [2] GIARRATANO, Joseph C. , RILEY, Gary D. . *Expert Systems: Principles and Programming, Fourth Edition*. [s.l.] : Course Technology, 2004. 842 s. ISBN 978-0534384470.

Tabulka [1] PROVAZNÍK, Ivo, KOZUMPLÍK Jiří. *Expertní systémy*. Vysoké učení technické v Brně, ústav biomedicínského inženýrství, fakulta elektrotechniky a informatiky, 1999.

Tabulka [2] HUSÁKOVÁ, Martina. *CLIPS A FUZZY CLIPS*, učební pomůcka k předmětu Znalostní technologie [online]. Univerzita Hradec Králové, fakulta informatiky a managementu, 2008. Dostupné na <http://lide.uhk.cz/fim/ucitel/fshusam2/index.html>.

8 Příloha

Kompletní kód expertního systému pro výpočet opravných položek k pohledávkám.

```
(defrule zadej_udaje
=>
  (printout t "Zadejte vysi pohledavky - oddelovac desetinnych mist je
tecka:" crlf)
  (bind ?castka (read))

  (printout t "Zadejte pocet mesicu po splatnosti nasledovne:" crlf)
  (printout t " 0 -  3 mesicu ..... zadejte 0" crlf)
  (printout t " 3 -  6 mesicu ..... zadejte 3" crlf)
  (printout t " 6 - 12 mesicu ..... zadejte 6" crlf)
  (printout t "12 - 18 mesicu ..... zadejte 12" crlf)
  (printout t "18 - 24 mesicu ..... zadejte 18" crlf)
  (printout t "24 - 30 mesicu ..... zadejte 24" crlf)
  (printout t "30 - 36 mesicu ..... zadejte 30" crlf)
  (printout t "nad  36 mesicu ..... zadejte 36" crlf)
  (bind ?inter (read))
  (while (not (or (eq ?inter 0) (eq ?inter 3) (eq ?inter 6) (eq ?inter
12) (eq ?inter 18) (eq ?inter 24) (eq ?inter 30) (eq ?inter 36)))
          (printout t "Mozne hodnoty 0,3,6,12,18,24,30 nebo 36! Zadejte
znovu:" crlf)
        (bind ?inter (read)))

  (printout t "Byl minuly rok pouzit §8c? (a/n):" crlf)
  (bind ?mr8c (read))
      (while (not (or (eq ?mr8c a) (eq ?mr8c n)))
          (printout t "Byl minuly rok pouzit §8c? (a/n):" crlf)
        (bind ?mr8c (read)))

  (printout t "Lze tvorit danovou opravnou polozku? (a/n):" crlf)
  (bind ?doplze (read))
      (while (not (or (eq ?doplze a) (eq ?doplze n)))
          (printout t "Lze tvorit danovou opravnou polozku? (a/n):" crlf)
        (bind ?doplze (read)))

  (printout t "Je vedeno soudni, rozhodci ci správni rizeni, jehoz se
radne ucastnite? (a/n):" crlf)
  (bind ?soud (read))
      (while (not (or (eq ?soud a) (eq ?soud n)))
          (printout t "Je vedeno soudni ci insolvencni rizeni? (a/n):"
crlf)
        (bind ?soud (read)))

  (printout t "Je pohledavka radne prihlasena v insolvencnim rizeni?
(a/n):" crlf)
  (bind ?insol (read))
      (while (not (or (eq ?insol a) (eq ?insol n)))
          (printout t "Je pohledavka radne prihlasena v insolvencnim
rizeni? (a/n):" crlf)
        (bind ?insol (read)))


  (assert (castka ?castka))
```

```
  (assert (inter ?inter))
  (assert (mr8c ?mr8c))
  (assert (doplze ?doplze))
  (assert (soud ?soud))
  (assert (insol ?insol))
  (assert (pomocuop 0)))


(defrule pravidlo1     ;nelze-li tvorit danovou op, je dopar rovno 0
  (declare (salience 5))
  (doplze n)
=>
  (assert (dopar 0)))



(defrule pravidlo6     ;vyssi priorita - zada-li uzivatel, ze lze tvorit
                       ;danovou op, ale interval je 0 nebo 3, pak
                       ;se vymaze moznost danove op zadana uzivatelem a
zada se, ze danova op nelze
  (declare (salience 10))
  (doplze a)
  (inter 0|3)
  (insol n)
   ?adr <- (doplze ?)
=>
  (retract ?adr)
  (assert (doplze n))
  (assert (dopar 0)))

(defrule pravidlo7     ;tvorba pravidla danove_bez_soudu
  (doplze a)
  (soud n)
  (insol n)
=>
  (assert (danove_bez_soudu a)))

;----------------------- insolvencni rizeni -----------------

(defrule insolvence
  (declare (salience 3))
  (insol a)
=>
  (assert (paraar 8))
  (assert (dopar 100)))

;----------------------- danova dle §8c ------------------

(defrule paragraf8czminula   ;byl-li min.rok pouzit §8c,bude i letos
  (declare (salience 2))
  (mr8c a)
  (insol n)
=>
  (assert (paraar 8c)))

(defrule paragraf8cnove      ;splni-li se podminky, doporuci se §8c
  (declare (salience 1))
  (castka ?castka)
  (test (<= ?castka 30000))
```

```
  (doplze a)
  (not (inter 0|3|6))
  (not (mr8c a))
  (not (insol a))
=>
  (assert (doporuc8c a)))

(defrule paragraf8c
  (paraar 8c)
=>
  (assert (dopar 100)))


;----------------------danove OP-------------------------


(defrule pravidlo8                ;podminky pro §8a1
  (danove_bez_soudu a)
  (castka ?x)
  (test (<= ?x 200000))
  (mr8c n)
  (insol n)
=>
  (assert (paraar 8a1)))


(defrule pravidlo9
  (paraar 8a1)
  (insol n)
=>
  (assert (dopar 20)))

(defrule pravidlo10
  (doplze a)
  (soud a)
=>
  (assert (danove_soud a)))

(defrule pravidlo11               ;podminky pro §8a1
  (danove_soud a)
  (castka ?x)
  (test (<= ?x 200000))
  (inter 6)
  (mr8c n)
=>
  (assert (paraar 8a1)))

(defrule pravidlo12               ;podminky pro §8a2
  (danove_soud a)
  (castka ?x)
  (test (<= ?x 200000))
  (inter ~6)
  (mr8c n)
=>
  (assert (paraar 8a2)))

(defrule pravidlo13
  (paraar 8a2|8a3)
  (inter 12)
```

```
=>
  (assert (dopar 33)))

(defrule pravidlo14
  (paraar 8a2|8a3)
  (inter 18)
=>
  (assert (dopar 50)))

(defrule pravidlo15
  (paraar 8a2|8a3)
  (inter 24)
=>
  (assert (dopar 66)))

(defrule pravidlo16
  (paraar 8a2|8a3)
  (inter 30)
=>
  (assert (dopar 80)))

(defrule pravidlo17
  (paraar 8a2|8a3)
  (inter 36)
=>
  (assert (dopar 100)))

(defrule pravidlo18                ;podminky pro §8a3
  (danove_soud a)
  (castka ?x)
  (test (> ?x 200000))
  (mr8c n)
=>
  (assert (paraar 8a3)))

(defrule pravidlo19
  (paraar 8a3)
  (inter 6)
=>
  (assert (dopar 20)))



;----------------------ucetni ----------------------------

(defrule pravidlo2     ;je-li interval 0, nelze tvorit ani ucetni OP
  (dopar 0)
  (inter 0)
=>
  (assert (uopar 0)))

(defrule pravidlo3     ;je-li interval 3, tvori se ucetni OP 25%
  (dopar 0)
  (inter 3)
=>
  (assert (uopar 25)))
```

```
(defrule pravidlo4           ;je-li interval roven 6 a nelze-li danova
                             ;OP, tvori se jen ucetni OP 50%
  (dopar 0)
  (inter 6)
=>
  (assert (uopar 50)))

(defrule pravidlo5           ;nelze-li danova OP, ale pohledavka je nad
                             ;12 mesicu, tvori se ucetni OP 100%
  (dopar 0)
  (inter ?inter)
  (test (>= ?inter 12))
=>
  (assert (uopar 100)))

(defrule pravidlouop1        ;dopocet ucetni OP do 50, resp. 100%
  (dopar 20)
  (inter ?inter)
=>
  (if (= ?inter 6)
    then (assert (uopar 30))
    else (assert (uopar 80))))


(defrule pravidlouop2        ;dopocet ucetni OP do 100%
  (dopar ?x)
  (pomocuop ?y)
  (test (or (eq ?x 33) (eq ?x 50) (eq ?x 66) (eq ?x 80) (eq ?x 100)))

=>
  (assert (uopar (- 100 ?x))))

;----------------------vypisy-------------------------------

(defrule vypis1
  (dopar ?dopar)
  (test (neq ?dopar 0))
  (paraar ?paraar)
  (uopar ?uopar)
 =>
  (printout t "Danova OP ve vysi " ?dopar "% vytvorena podle §" ?paraar
". Ucetni OP je ve vysi " ?uopar "%." crlf))

(defrule vypis2
  (doporuc8c a)
 =>
  (printout t "Lze doporucit 100% danovou OP dle §8c do vyse max. 30000
Kc na dluznika." crlf))


(defrule vypis3
  (dopar 0)
  (dopar ?dopar)
  (uopar ?uopar)
 =>
  (printout t "Danova OP nemohla byt vytvorena. Ucetni OP je ve vysi "
?uopar "%." crlf))
```