

PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO  
KATEDRA INFORMATIKY

## BAKALÁŘSKÁ PRÁCE

Implementace expertního systému s dopředným řetězením

Forward-chaining Expert System Implementation



2010

Jakub Kaláb

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně.

5.8.2010

Jakub Kaláb

## Anotace

*Expertní systémy mají v praxi bohaté využití. Jejich smyslem je asistovat expertovi na danou problematiku, či jej plně nahradit. V příloze bakalářské práce implementuji prázdný expertní systém s dopředným řetězením inspirovaný systémem CLIPS jako knihovnu v programovacím jazyku Common Lisp tak, aby jej bylo možno plně integrovat do dalších programů.*

Děkuji Mgr. Martinu Dostálovi, PhD. za vedení této bakalářské práce.

# Obsah

<b>1. Co je to expertní systém</b>	<b>1</b>
1.1. Co odlišuje expertní systém od jiných výpočetních programů . . .	1
1.2. Charakteristické vlastnosti expertních systémů . . . . .	1
<b>2. Uživatelská příručka</b>	<b>2</b>
2.1. Použité datové struktury . . . . .	2
2.2. Manipulace s fakty znalostní baze . . . . .	3
2.2.1. Definice skupiny faktů . . . . .	3
2.3. Manipulace s odvozovacími pravidly . . . . .	3
2.4. Spouštění inferenčního mechanismu . . . . .	4
<b>3. Použité algoritmy</b>	<b>5</b>
<b>4. Popis implementace</b>	<b>6</b>

# **1. Co je to expertní systém**

Expertní systém je počítačový program, který ze zadané znalostní báze pomocí zadaných pravidel odvozuje nová fakta. Díky tomu je jej možno využít v praxi jako asistenta odborníka v daném oboru, nebo jej, v ideálním případě, zcela nahradit.

Pro lepší představu uvedu příklad - expertní systém v ordinaci praktického lékaře. Na začátku zadá lékař, případně za pomoci znalostního inženýra, expertnímu systému znalostní bazi, tj. informace o příznacích známých chorob, možnostech jejich léčby, medikaci, konfliktních léků, atd. Expertní systém potom při zadání příznaků pacienta s jistou pravděpodobností (odvislé od pravděpodobností zadaných v jednotlivých pravidlech příznak  $\rightarrow$  choroba) určí možné choroby, alternativy léčby, apod.

## **1.1. Co odlišuje expertní systém od jiných výpočetních programů**

Toto samozřejmě může dělat i program, který bychom za expertní systém neoznačili, rozdíl je v tom, že expertní systém je obecný - znalostní báze je zcela oddělena řídicího mechanismu. Tento je tudíž zcela nezávislý na konkrétní doméně. Takto tomu u jiných programů většinou nebývá, data bývají smýchána s rozhodovacím a řídicím kódem programu a program je tudíž jednoúčelový.

## **1.2. Charakteristické vlastnosti expertních systémů**

## 2. Uživatelská příručka

Tato práce se zabývá implementací již známých algoritmů, je tedy spíše praktického zaměření. Úmyslně jsem tedy posunul kapitolu s hlubším popisem implementovaných algoritmů až za uživatelskou příručku, aby byl čtenář při její četbě již seznámen s používanými datovými strukturami, strukturou pravidel, apod.

Tato kapitola vyžaduje alespoň zběžnou znalost programovacího jazyka Common Lisp, jeho základní datové typy (symbol, seznam) a metody pro práci s nimi.

V kapitole budu občas uvádět v závorkách a kurzívou názvy některých použitých tříd či metod. Je to kvůli snadnější orientaci při následné četbě programátorské dokumentace. Čtenář, který tuto číst nehodlá je může směle přeskakovat.

### 2.1. Použité datové struktury

Uživatel expertního systému se nejčasteji setká se dvěma datovými strukturami, jsou to fakta (třída *fact*) a pravidla (třída *rule*). Fakta znalostní baze mohou být buď jednoduchá (třída *simple-fact*), specifikovaná prostým seznamem atomů (např. (on book table)), či strukturovaná (třída *template-fact*. Pro použití strukturovaných fakt je třeba nejdříve definovat šablonu makrem `deftemplate` následovně:

```
(deftemplate in (object location (ammount :default 1)))
```

Specifikace takového strukturovaného faktu pak vypadá takto: (in :object fridge :location kitchen).

Pozn.: Při opakovaném volání `deftemplate` se stejným jménem šablony dochází k přepsání původní definice. Má-li nově definovaná šablona jiné sloty než šablona původní a existují-li ve znalostní bazi fakta vytvořená pomocí původní šablony, vzniká nekonzistence. Makro je tedy třeba volat s rozmyslem

Kromě fakt se uživatel setká ještě s tzv. *patterny* (pro neznalost výstižného českého ekvivalentu se budu držet dobře zažitého výrazu anglického). Tyto jsou faktům velmi podobné, jen se v nich mohou vyskytovat proměnné. Název proměnné v mé implementaci vždy začíná otazníkem (stejně jako v systému CLIPS) a je tedy od běžného atomu jasně patrná. Patterny mohou být, stejně jako fakta jednoduché či strukturované. Při snaze použít proměnnou v popisu faktu skončí vyhodnocení výrazu chybou.

Převodní pravidlo expertní systému se skládá ze dvou částí - podmínek a aktivací. Podmínky určují, za jakých okolností je pravidlo splněno fakty znalostní baze a je jej možno zařadit do seznamu pravidel k aktivaci (tento seznam budu dále označovat jako *agenda*). Seznam aktivací je tvořen libovolným počtem Lispových výrazů (jež pochopitelně mohou zahrnovat i systémem definované metody a makra), které se při aktivaci pravidla vyhodnotí. Pravidlo tedy typicky při platnosti nějakých fakt či neplatnosti jiných přidá do znalostní baze nějaká další fakta, či některá z baze odebere.

## 2.2. Manipulace s fakty znalostní baze

K přidání faktu do znalostní baze souží makro **assert**, přidání tedy vypadá následovně:

```
(assert (on book table))  
(assert (in :object fridge :location kitchen))
```

Při pokusu o opětované přidání již existujícího faktu se nestane nic.

Odebrání faktu z baze se provádí makrem **retract** se stejnou syntaxí. Při pokusu o odebrání neexistujícího faktu se opět nic nestane.

Posledním makrem z této skupiny je **modify** to slouží k modifikaci faktu (jeho nahrazení jiným) a používá se takto:

```
(modify (in :object snack :location fridge)  
        (in :object snack :location schoolbag))
```

Toto makro je ve skutečnosti jen zkratkou pro postupné volání **assert** a **retract**. Neslouží ani tak k ušetření zdrojového kódu, jako spíš ke zčitelnění aktivací pravidla - z jeho použití je evidentní vztah dvou faktů.

### 2.2.1. Definice skupiny faktů

Pro pohodlnější práci s větším množstvím faktů jsem (stejně jako v systému CLIPS) implementoval makro **deffacts**. Volání tohoto makra obsahuje název skupiny faktů následovaný požadovanými fakty:

```
(deffacts stuff-i-ve-got-in-me-fridge  
  (in :object coke :location fridge)  
  (in :object cheese :location fridge)  
nn  (is-expired cheese)  
  (in :object butter :location fridge)  
  ...  
)
```

Takto definovaná fakta se ve znalostní bazi neobjeví hned, nýbrž až po zavolání makra **reset** (bez parametrů). Toto makro odstraní ze znalostní baze všechna fakta a poté znalostní bazi naplní fakty definovanými ve skupinách makrem **deffacts**. K pouhému vyčištění znalostní baze slouží makro **clear** (opět bez parametrů).

## 2.3. Manipulace s odvozovacími pravidly

Odvozovací pravidla expertního systému se skládají ze dvou částí - seznamu podmínek a seznamu aktivací. Podmínky určují, jaká fakta musí platit, aby bylo



možno pravidlo aplikovat. Seznam aktivací pak obsahuje libovolný počet lis-pových výrazů, které jsou při aplikaci vyhodnoceny. Tyto samozřejmě mohou (a velmi často budou) zahrnovat i makra a metody definované touto knihovnou.

Definice odvozovacího pravidla se provádí makrem **defrule** a zahrnuje jméno pravidla a seznamy podmínek a aktivací. Podmínky jsou od aktivací odděleny symbolem **=>**. Vypadá to tedy nějak takto:

```
(defrule find-and-take-green-fruit-to-the-left-of-orange-vegetable
  (green ?x)
  (fruit ?x)
  (orange ?y)
  (vegetable ?y)
  (next-to :left-one ?x :right-one ?y)
=>
  (take ?x)
  (take ?y))
```

Při opakovaném volání **defrule** se stejným jménem pravidla dochází k přepsání jeho definice.

K zneplatnění definice pravidla slouží makro **undefrule**, jehož jediným parametrem je název pravidla.

## 2.4. Spouštění inferenčního mechanismu

### 3. Použité algoritmy

Příložená implementace expertního systému zahrnuje několik z literatury čerpaných algoritmů. V první řadě je to algoritmus Rete, který se stará o zjišťování, která odvozovací pravidla jsou splněna fakty ze znalostní baze. Poté je použito několik velice jednoduchých algoritmů pro výběr pravidla, jenž bude jako další aktivováno.

## 4. Popis implementace