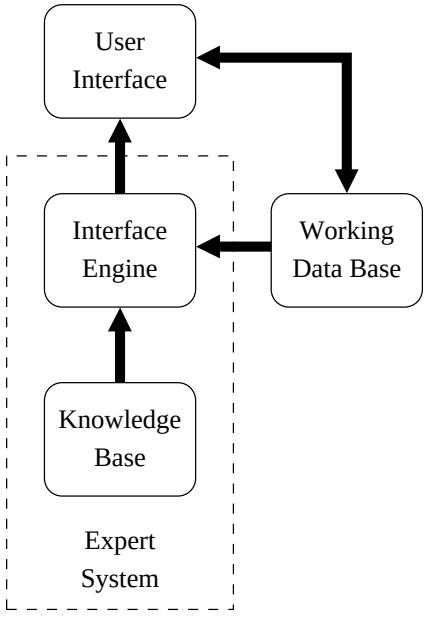
1 Expertní systémy

Expertní systémy jsou jednou z komerčně nejúspěšnějších oblastí umělé inteligence. Charakterizovat stručně tuto rozsáhlou oblast není jednoduché. Pokusíme se však o to alespoň následovně:

„Expertní systémy představují třídu počítačových programů, které mohou poradit, analyzovat, klasifikovat, sdělit, konzultovat, navrhovat, diagnostikovat, vysvětlit, hledat, předpovídat, formovat návrhy, identifikovat, interpretovat, formovat, učit, organizovat, monitorovat, plánovat, prezentovat, vydolovat, stanovovat, testovat a vést. Jsou určeny pro takové problémy, které pro své uspokojivé řešení vyžadují experta z daného oboru.“

Podívejme se teď na složení expertního systému podrobněji. Blokové schéma vidíme na obrázku 1. Systém se skládá ze čtyř základních částí. Jádro expertního systému je tvořeno **řídícím modulem** a **bází znalostí**, kam ukládáme znalosti expertů. Řídící modul obsahující inferenční mechanismus, pak interpretuje a prochází uložené znalosti. Tento modul ale musí ke znalostem připojit i **pracovní data**, která popisují konkrétní, právě řešenou situaci, nebo problém. Posledním modulem ES je **uživatelské rozhraní**, které musí srozumitelnou formou komunikovat s uživatelem, ukládat získané informace do pracovní databáze a interpretovat odvozené informace.



Obrázek 1: Základní moduly experního systému

BZ	Počet produkčních pravidel	člověkoroky
malá	50-350	$1/4 - 1/2$
velká	500-3000	1 - 2
velmi velká	10000	3 - 5

Tabulka 1: Orientační časová náročnost tvorby BZ

1.1 Báze znalostí

Báze znalostí je tou částí ES, kde jsou uloženy znalosti expertů. Je proto kladen velký důraz na reprezentaci těchto znalostí, aby získané znalosti nebyly znehodnoceny. V této oblasti probíhá neustálý výzkum a tato problematika patří mezi stěžejní témata umělé inteligence.

Způsobů, jak reprezentovat poznatky je mnoho, každý má své klady i zápory. Máme-li tedy uvažovat o obecných charakteristikách reprezentace poznatků, musíme vzít v úvahu minimálně tři následující pohledy:

vyjadřovací účinost použitých reprezentačních prostředků, tedy co a do jaké míry jimi lze reprezentovat,

odvozovací účinnost prostředků, tedy co a jak jimi lze odvozovat,

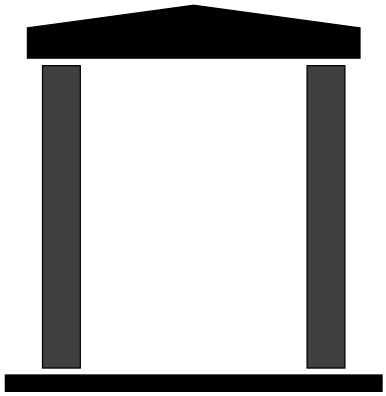
výpočetní efektivita prostředků, t.j. složitost podprogramů pro práci se symboly v podmínkách.

Dá se říci, že tyto pohledy korespondují s vnitřním, vnějším a implementačním pohledem na bázi znalostí a měly by velmi úzce souviset s obdobnými úvahami při implementaci inferenčního mechanismu, který s danou bází znalostí bude pracovat.

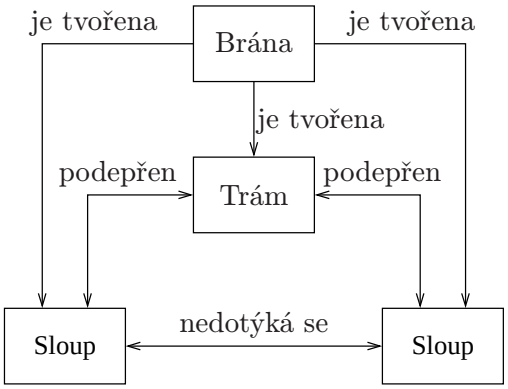
Samotné sestavování báze znalostí je pak poměrně dlouhodobá záležitost. A protože panují trochu zkreslené představy o tom, jak dlouho taková BZ vzniká, je dobré si udělat alespoň hrubou představu o čase potřebném pro získávání poznatků od expertů. Orientační údaje jsou v tabulce 1. Tyto údaje se ale mohou lišit případ od případu, téma od tématu.

1.2 Reprezentace báze znalostí

Zatím se nepodařilo stanovit všeobecně platná kritéria pro volbu nejvhodnějšího způsobu reprezentace báze znalostí. Poznatky expertů jsou totiž velmi různorodé a můžeme je rozdělit do několika skupin. Podle těchto skupin pak rozlišujeme i způsoby reprezentace BZ. Následující výčet ale určitě nebude úplný. Jde pouze o naznačení různorodosti možných poznatků.



Obrázek 2: Jednoduchá skladba objektů - brána



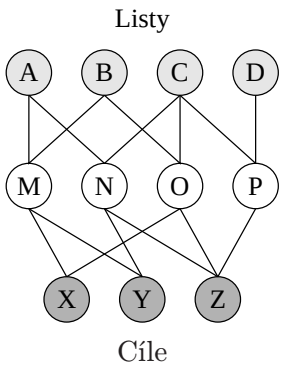
Obrázek 3: Vztahy mezi objekty brány

1.2.1 Asociativní

Odpovídají obvyklým, odpozorovaným, nebo pravděpodobným vztahům mezi dvěma objekty. Vztahy však nemusí být zdůvodnitelné.

Řadu informací kolem nás musíme brát tak, jak jsou. Jako příklady můžeme vzít světové strany - sever, jih, východ a západ. Planety sluneční soustavy, fyzikální vlastnosti látek a mohli bychom hledat další nezdůvotnitelné příklady.

Jak mohou vypadat vztahy odpozorované a obvyklé, si ukážeme na jednoduchém příkladu. Mějme jednoduchou konstrukci brány, jako na obrázku 2. Vidíme tři jednoduché stavební prvky - dvě podpěry nesoucí vodorovný trám. Vztahy mezi objekty si pak můžeme znázornit na obrázku 3.



Obrázek 4: Příklad inferenční sítě

1.2.2 Kauzální

Odpovídají známým a zdůvodnitelným vztahům mezi dvěma objekty, které jsou, nebo mohou být ve vztahu příčiny a důsledku.

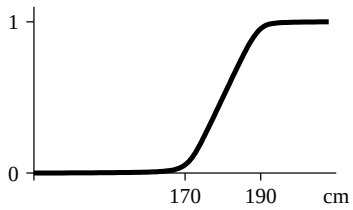
Jako příklad těchto znalostí mohou být takzvané **inferenční sítě**. Jde o docela obvyklou formulaci znalostí ve formě „pokud je splněna podmínka, pak z toho plyne důsledek“. Sestavme si jednoduchý příklad:

```
if A and B then M
if A or C then N
if B and not C then O
if C and D then P
if M or O then X
if M and N then Y
if N or O and P then Z
```

Graficky si můžeme tuto síť znázornit na obrázku 4. Velkými písmeny označujeme jednotlivé **hypotézy** a z obrázku je patrné, že všechny hypotézy lze rozdělit do tří skupin. První skupinu tvoří takové hypotézy, které nejsou odvoditelné. V našem příklady jsou to hypotézy A, B, C, D a nazýváme je **listové**. Druhou skupinou jsou hypotézy X, Y, Z, ze kterých už žádný další důsledek neplyne a proto jim říkáme **cílové**. Ostatní hypotézy označujeme za **mezilehlé**.

Pravděpodobnostní přístup

Odvozovací pravidla nemusí být jen jako dvoustavová logika. Mužeme každé pravidlo rozšířit o pravděpodobnost, s jakou budou potvrzeny jednotlivé hypotézy a z toho bude plynout s jistou pravděpodobností i důsledek. Pro vyhodnocování takových pravidel využíváme Bayesovu formuli pro výpočet podmíněné pravděpodobnosti ve tvaru:



Obrázek 5: Pravděpodobnost, že člověk je vysoký

$$P(d|p) = \frac{P(p|d).P(d)}{P(p)}$$

Význam jednotlivých členů ve vzorci je následující:

p podmínka pravidla,

d důsledek pravidla,

$P(d|p)$ podmíněná pravděpodobnost d při splnění p,

$P(p|d)$ podmíněná pravděpodobnost p při splnění d,

$P(p)$ pravděpodobnost jevu p,

$P(d)$ pravděpodobnost jevu d.

Abychom mohli sestavit bázi znalostí rozšířenou o pravděpodobnostní přístup, musíme mít k dispozici dostatečně důvěryhodný soubor vzorků, ze kterých pravděpodobnosti spočítáme. Bez dobrých podkladů však nemá pravděpodobnostní přístup smysl!

Použití fuzzy logiky

Dalším způsobem rozšíření dvoustavové logiky na vícehodnotovou, je fuzzy logika. Pravdivostní hodnota každé hypotézy se může měnit spojitě v rozmezí $\langle 0, 1\rangle$. Tento stupeň pravdivosti nazýváme ve fuzzy logice plauzabilitou - věrohodností. Obvykle stanovujeme pro každou hypotézu převodní funkci, ale můžeme také převod realizovat jen tabulkou. Příklad převodní funkce, kdy chceme vyjádřit, zda je někdo *vysoký člověk*, je na obrázku 5.

Fuzzy logiku můžeme snadno použít v inferenčních sítích. Musíme ale umět stanovit věrohodnost hypotéz, které jsou obsaženy ve složených podmínkách odvozovacích pravidel. V případě konjunkce více hypotéz můžeme jejich věrohodnost stanovit následovně:

$$V(\wedge(t_1, t_2, t_3, ..., t_n)) = min(V(t_1), V(t_2), V(t_3), ..., V(t_n))$$

U konjunkce tedy vybíráme jako výslednou věrohodnost minimální hodnotu z věrohodností jednotlivých hypotéz. Podobně můžeme stanovit vyhodnocení disjunkce:

$$V(\vee(t_1, t_2, t_3, ..., t_n)) = max(V(t_1), V(t_2), V(t_3), ..., V(t_n))$$

A jako poslední musíme umět vyhodnotit negaci:

$$V(\bar{t}) = 1 - V(t)$$

Vyhodnocovat složené podmínky tedy umíme a teď už jen uvedeme, jak stanovíme věrohodnost důsledku odvozovacího pravidla:

$$V(d) = V(p).V(r)$$

kde d je věrohodnost důsledku,

p věrohodnost podmínky pravidla daná libovonou logickou funkcí,

r věrohodnost pravidla.

1.2.3 Kontextové

Odpovídají podmínkám, při kterých se uplatňují různé souvislosti mezi objekty.

Z praxe asi každý zná mnoho situací, kdy určitá informace závisí na kontextu, ve kterém je použita. Jednoduchým příkladem by mohla být informace o vlastním zdravotním stavu, kdy řekneme: „měl jsem zvýšenou teplotu“. Všichni víme, že zvýšená teplota může být důsledkem, předzvěstí, ale i průvodním jevem jakékoliv zdravotní indispozice. Mužeme tedy říci, že jde o informaci *kontextově* závislou.

1.2.4 Prostorové

Odpovídají prostorovým vztahům mezi objekty. A to nejen ve smyslu polohy, ale i směru, orientace, nebo rychlosti.

Příkladem by mohly být informace potřebné pro řízení letového provozu ve vzdušném prostotu každé země. Podle typu letadla, jeho kurzu a rychlosti musíme stanovovat jeho letovou hladinu.

1.2.5 Taxonomické

Odpovídají známým souvislostem mezi objekty, které vystihují postup od obecného ke speciálnímu, tzv. generalizování. Jako příklad se můžeme podívat, jak bychom tuto metodu použili pro popis jednoduché stavby, v našem případě garáž:

- garáž
    - severní stěna
        - cihly
        - omítka
        - větrací mřížka
    - východní stěna
        - cihly
        - omítka
    - jižní stěna
        - cihly
        - omítka
        - vrata
            - zárubně
            - křídla
            - panty
            - zámek
    - západní stěna
        - cihly
        - omítka
        - okno
            - rám
            - okno
            - kování
    - střecha
        - trámy
        - desky
        - krytina
        - oplechování
            - okapnice
            - rýny
            - svody
    - podlaha
        - beton
        - izolace
        - potěr

Celou stavbu rozebíráme na menší celky, ty dále opět dělíme na menší části a pokračujeme v rozboru až na díly, které z hlediska stavby považujeme za již dále nedělitelné.

1.2.6 Časové

Odpovídají takovým souvislostem mezi objekty, kdy jejich změna v určitých časových okamžicích podmiňuje okamžité, nebo časově vázané změny nebo výskyty jiných objektů. Dovoluje uvažovat o tom co bylo, je a bude.

Mezi systémy, kde hraje důležitou roli časová souvislost, můžeme zařadit např. sestavování jízdních řádů, kde musíme zohlednit počet vozidel, souprav, řidičů a jejich pracovní režim. Dalším příkladem by mohlo být plánování jízd vlakových souprav, kde kromě pravidelných linek osobní dopravy se po kolejích pohybují nepravidelné nákladní vlaky. A musíme se vyvarovat situace, kdy bychom si nevhodným časovým plánem jízd mohli zaplnit některé nádraží a ocitnout se v patové situaci (je zde jistá podobnost s notoricky známou hrou sokoban).

1.2.7 Modelové

Odpovídají známým nebo předpokládaným souvislostem mezi více provázanými objekty, které se ovlivňují a tím určují chování systému.

Asi nejnázornějším příkladem modelových znalostí může být pro většinu populace autoškola a křižovatky. Na modelových situacích se každý řidič učí rozhodovat o pořadí průjezdu všech účastníků silničního provozu křižovatkou a pak musí tyto modelové situace aplikovat v praxi. Úkolem expertů je, vhodné modelové příklady sestavit.

1.3 Inferenční mechanismy

Podíváme-li se na bázi znalostí reprezentovanou inferenční sítí, jako to je na obrázku 4, vidíme její rozklad na 3 množiny hypotéz. Množinu listů, cílů a mezilehlých hypotéz. Pokud máme takto zadanou bázi znalostí, přichází automaticky důležitá otázka: k čemu ji použijeme? Jsou zde dvě reálné možnosti.

Ocitli jsme se v situaci kdy potřebujeme poradit jak dál. Zajímá nás, jaké důsledky pro nás může stávající situace mít. Chceme tedy postupovat od **listů** k **cílům**.

Druhá možnost je obrácená. Ocitli jsme se v nějaké situaci a budeme se znažit zjistit, co tuto situaci mohlo způsobit. Zajímají nás možné příčiny atuálního stavu. Budeme proto postupovat od **cílů** k **listům**.

Podle směru procházení inferenční sítě se také odvíjí názvy dvou možný inferenčních mechanismů - **dopředné** a **zpětné** řetězení.

1.3.1 Zpětné řetězení

Princip tohoto algoritmu byl nastíněn. Úkolem je procházet inferenční síť od cílů směrem k listům. Než ale začneme algoritmus popisovat, musíme si vytvořit vlastní bázi znalostí. Nebudeme si ovšem dávat žádné velké cíle a snažit se vyrovnat expertům z libovolného oboru. Naše snaha se musí soustředit na jednoduchost a srozumitelnost. Mějme tedy například následující bázi znalostí:

```
:-op( 200, fx, cil ).
:-op( 200, fx, list ).
:-op( 200, fx, if ).
:-op( 100, xfx, then ).
:-op( 50, xfy, and ).
:-op( 50, xfy, or ).

cil savec.
cil ptak.
cil selma.
cil dravec.
cil bylozravec.

list saje_mleko.
list ma_peri.
list leta.
list ma_drapy.
list zere_maso.
```

```
if saje_mleko then savec.
if neg( savec ) and ( ma_peri or leta ) then ptak.
if savec and ( ma_drapy or zere_maso ) then selma.
if ptak and ( ma_drapy or zere_maso ) then dravec.
if savec and neg( selma ) then bylozravec.
```

Pro dobrou přehlednost a srozumitelnost jsme využili navíc možnosti Prologu a definovali jsme si operátory: `cil, list, if, then, and, or`. Můžeme tak zapisovat všechny informace v čitelnější infixové notaci.

Vytvořili jsme celkem 5 odvozovacích pravidel, jejichž obsah asi nevyžaduje další komentáře. Máme v nich 10 hypotéz, z toho jich je 5 listových a 5 cílových a jsou vypsány před odvozovacími pravidly. Podrobnějším zkoumáním sice zjístíme, že mezi cílové hypotézy jsme pro zjednodušení zařadili i dvě mezilehlé hypotézy. Toto zjednodušení si však v této chvíli můžeme dovolit. (Jen tak pro kontrolu, víte které jsou ty mezilehlé hypotézy?)

Máme tedy odvozovací pravidla v naší bázi znalostí a můžeme začít řešit samotný inferenční mechanismus. Jako první krok bude vhodné, naučit se vyhodnocovat podmínky. Predikát řešící tento úkol může vypadat následovně:

```
podminka( Hyp ) :- % 1
    poz_fakt( Hyp ), !.

podminka( Hyp ) :- % 2
    neg_fakt( Hyp ), !, fail.

podminka( neg( Hyp ) ) :- % 3
    podminka( Hyp ), !, fail; !.

podminka( Hyp ) :- % 4
    list Hyp, !, dotaz( Hyp ).

podminka( Hyp1 and Hyp2 ) :- % 5
    !, podminka( Hyp1 ), podminka( Hyp2 ).

podminka( Hyp1 or Hyp2 ) :- % 6
    podminka( Hyp1 ), !; podminka( Hyp2 ).
```

Vysvětleme si jednotlivé kroky:

1. Ověření, zda je hypotéza mezi dříve splněnými fakty.
2. Hypotéza by také už mohla být mezi dříve nesplněnými fakty.
3. V případě negace hypotézy musíme výsledek ověření negovat.

4. Listovou hypotézu nelze nijak odvodit a musíme se dotázat uživatele.

5. Vyhodnocení složených podmínek. Konjunkce.

6. Disjunkce.

K predikátu `podminka` si ještě přidáme 2 pomocné predikáty. Jeden pro úklid dříve zodpovězených hypotéz. Druhý pak bude pokládat dotazy uživateli:

```
uklid :-
    retractall( poz_fakt( _ ) ),
    retractall( neg_fakt( _ ) ).

dotaz( Hyp ) :-
    write( '? Je pravda, ze ' ),
    write( Hyp ),
    write( ' ? [a/n] ' ),
    read( Odp ),
    Odp = a,
    assert( poz_fakt( Hyp ) ).
dotaz( Hyp ) :-
    assert( neg_fakt( Hyp ) ), fail.
```

Všechny pomocné predikáty jsou již k dispozici a můžeme se podívat, jak bude vypadat samotný algoritmus zpětného řetězení.

```
krok_zpet( Hyp ) :-
    if Podm then Hyp,
    podminka( Podm ),
    assert( poz_fakt( Hyp ) ).

zpet( Hyp ) :-
    write( '> Bude overovana hypoteza: ' ),
    write( Hyp ), nl,
    krok_zpet( Hyp ),
    write( '+ Hypoteza ' ),
    write( Hyp ),
    write( ' splnena!' ), nl, !.
zpet( Hyp ) :-
    write( '- Hypoteza ' ),
    write( Hyp ),
    write( ' neni splnena!' ), nl, !, fail.
```

Jednoduché? Ano, predikát `zpet` je jen pomocným predikátem. Vypisuje informace o průběhu řešení uživateli. Samotný problém zpětného řetězení

řeší jen predikát `krok_zpet`. Ten se pokusí pro zadanou hypotézu najít takové odvozovací pravidlo, ve kterém je požadovaná hypotéza jako důsledek. Bude-li takové pravidlo nalezeno, musí být splněna jeho podmínka, aby se hypotéza potvrdila.

Postupujeme tedy od důsledku k příčině, od cíle k listům. Provádíme **zpětné řetězení**.

Navržený systém můžeme ještě doplnit o jeden predikát, kterým vytvoříme velmi jednoduché uživatelské rozhraní. Budeme se „naslepo“ dotazovat uživatele na vlastnosti jednoho živočicha a pokusíme se jej klasifikovat.

```
start_zpet :-
    uklid,
    cil Hyp,
    write( '> Vybrany cil je "' ),
    write( Hyp ),
    write( '".' ), nl,
    zpet( Hyp ),
    write( '? Dalsi cil? [a/n]: ' ),
    read( Odp ),
    Odp = n,
    write( 'Konec...' ),
    nl, !.
start_zpet :-
    write( '> Nejsou dalsi cile.' ), nl, !.
```

Nyní si vybereme tři živočichy, na kterých si ověříme, zda je navžený systém bude schopen správně klasifikovat. Mějme tedy např. zajíce, havrana a lva. Začneme zajícem. Náš expertní systém spustíme predikátem `start_zpet`.

```
?- start_zpet.
> Vybrany cil je "savec".
> Bude overovana hypoteza: savec
? Je pravda, ze saje_mleko ? [a/n] a.
+ Hypoteza savec splnena!
? Dalsi cil? [a/n]: a.
> Vybrany cil je "ptak".
> Bude overovana hypoteza: ptak
- Hypoteza ptak neni splnena!
> Vybrany cil je "selma".
> Bude overovana hypoteza: selma
? Je pravda, ze ma_drapy ? [a/n] n.
? Je pravda, ze zere_maso ? [a/n] n.
```

```
- Hypoteza selma neni splnena!
> Vybrany cil je "dravec".
> Bude overovana hypoteza: dravec
- Hypoteza dravec neni splnena!
> Vybrany cil je "bylozravec".
> Bude overovana hypoteza: bylozravec
+ Hypoteza bylozravec splnena!
? Dalsi cil? [a/n]: a.
> Nejsou dalsi cile.
Yes
```

Z výpisu vidíme chování systému. Řádky začínající znakem „?“ jsou dotazy pro uživatele. Znakem „>“ jsou označeny oznámení pro uživatele a zbývající dva znaky „+“ a „-“ označují splněné a nesplněné subcíle.

Pokud jsme zodpovědně odpovídali na otázky týkající se vlastností zajíce, provedl systém základní klasifikaci: je to savec a býložravec. S takovýmto výsledkem můžeme být spokojeni.

Teď zkusíme, jak si systém poradí s havranem.

```
?- start_zpet.
> Vybrany cil je "savec".
> Bude overovana hypoteza: savec
? Je pravda, ze saje_mleko ? [a/n] n.
- Hypoteza savec neni splnena!
> Vybrany cil je "ptak".
> Bude overovana hypoteza: ptak
? Je pravda, ze ma_peri ? [a/n] a.
+ Hypoteza ptak splnena!
? Dalsi cil? [a/n]: a.
> Vybrany cil je "selma".
> Bude overovana hypoteza: selma
- Hypoteza selma neni splnena!
> Vybrany cil je "dravec".
> Bude overovana hypoteza: dravec
? Je pravda, ze ma_drapy ? [a/n] n.
? Je pravda, ze zere_maso ? [a/n] a.
+ Hypoteza dravec splnena!
? Dalsi cil? [a/n]: a.
> Vybrany cil je "bylozravec".
> Bude overovana hypoteza: bylozravec
- Hypoteza bylozravec neni splnena!
> Nejsou dalsi cile.
Yes
```

Projdeme si řádky začínající „+“ a dostáváme informace, že havran je pták a současně dravec. Systém opět provedl správné hodnocení.

Poslední z naší trojice živočichů je lev. Bude i nyní chování systému správné?

```
?- start_zpet.
> Vybrany cil je "savec".
> Bude overovana hypoteza: savec
? Je pravda, ze saje_mleko ? [a/n] a.
+ Hypoteza savec splnena!
? Dalsi cil? [a/n]: a.
> Vybrany cil je "ptak".
> Bude overovana hypoteza: ptak
- Hypoteza ptak neni splnena!
> Vybrany cil je "selma".
> Bude overovana hypoteza: selma
? Je pravda, ze ma_drapy ? [a/n] a.
+ Hypoteza selma splnena!
? Dalsi cil? [a/n]: a.
> Vybrany cil je "dravec".
> Bude overovana hypoteza: dravec
- Hypoteza dravec neni splnena!
> Vybrany cil je "bylozravec".
> Bude overovana hypoteza: bylozravec
- Hypoteza bylozravec neni splnena!
> Nejsou dalsi cile.
Yes
```

I lev byl klasifikován správně. Podle zjištěných závěrů je to savec a šelma.

V této chvíli bychom mohli být s výsledkem spokojeni. Je ovšem třeba upozornit, že jsme ještě všechno nevyzkoušeli. Co kdybychom se pokusili použít přímo predikát `zpet` k ověření námi vybrané hypotézy? Mějme situaci, kdy nás zajímá, zda zajíc je býložravec. Uděláme tedy pokus a nesmíme zapomenout předem uklidit předchozí dotazy.

```
?- uklid.
Yes
?- zpet(bylozravec).
> Bude overovana hypoteza: bylozravec
- Hypoteza bylozravec neni splnena!
No
?-
```

Je určitě zajímavé, proč systém okamžitě naši hypotézu zamítnul a přitom uživateli nepoložil jedinou otázku! Není se ovšem čemu divit. Zapomněli jsme na jednu zásadní věc. Na rekurzi.

Celý problém pramení z predikátu `krok_zpet`. Jak už jeho samotný název napovídá, provádí zpětné řetězení, ale jen o jeden krok zpět. **Bude-li se však v podmínce vyskytovat jakákoliv hypotéza, která není listová, systém ji zpětným řetězením neověří!** Pro odstranění tohoto nedostatku stačí rozšířit predikát `podminka` o jedno pravidlo, kterým vytvoříme rekurzi.

```
podminka( Hyp ) :-
    cil Hyp, !, zpet( Hyp ).
```

Znovu můžeme zkusit, jak se systém bude chovat při snaze o ověření hypotézy, zda je zajíc býložravec.

```
?- uklid.
Yes
?- zpet(bylozravec).
> Bude overovana hypoteza: bylozravec
> Bude overovana hypoteza: savec
? Je pravda, ze saje_mleko ? [a/n] a.
+ Hypoteza savec splnena!
> Bude overovana hypoteza: selma
? Je pravda, ze ma_drapy ? [a/n] n.
? Je pravda, ze zere_maso ? [a/n] n.
- Hypoteza selma neni splnena!
+ Hypoteza bylozravec splnena!
Yes
```

Nyní už se zdá, že je opravdu vše v pořádku. Proč ovšem systém dříve ověřil zajíce, havrana i lva správně, zůstane čtenáři k zamyšlení.

1.3.2 Dopředné řetězení

Inferenční mechanismus založený na dopředném řetězení nezačneme budovat na zelené louce. Využijeme vše, co bylo vytvořeno a vyzkoušeno v předchozí kapitole o zpětném řetězení. K vytvořeným predikátům jen přidáme čtyři další.

Pro dopředné řetězení budeme potřebovat postupovat při vyhodnocování hypotéz od listů k cílům. Budeme proto nuceni hledat mezi odvozovacími pravidly taková, která mají náš předpoklad v podmínce. To není tak elementární úkol, jako hledání pravidel, kde je hypotéza přímo důsledkem. Vytvoříme si proto pomocný predikát, který bude ověřovat, zda je libovolná hypotéze obsažena ve složené podmínce.

```
vpodmince( Hyp, Hyp ) :- !.
vpodmince( Hyp, neg( Hyp ) ).
vpodmince( Hyp, A and B ) :-
    vpodmince( Hyp, A ); vpodmince( Hyp, B ).
vpodmince( Hyp, A or B ) :-
    vpodmince( Hyp, A ); vpodmince( Hyp, B ).
```

S tímto pomocným predikátem `vpodmince` můžeme vytvořit jádro inferenčního mechanismu pro dopředné řetězení. Realizace bude velmi podobná, jako u zpětného řetězení.

```
krok_dopredu( Hyp, Dusl ) :-
    if Podm then Dusl,
    vpodmince( Hyp, Podm ),
    podminka( Podm ),
    assert( poz_fakt( Dusl ) ).

dopredu( Hyp ) :-
    write( '> Hledame dusledek hypotezy: ' ),
    write( Hyp ), nl,
    krok_dopredu( Hyp, Dusl ),
    write( '+ Z hypotezy ' ),
    write( Hyp ),
    write( ' plyne dusledek: ' ),
    write( Dusl ), nl, !.
dopredu( Hyp ) :-
    write( '- Z hypotezy ' ),
    write( Hyp ),
    write( ' nebylo nic odvozeno.' ), nl, !, fail.
```

Slibovali jsme čtyři predikáty a zatím byly jen tři. Poslední predikát slouží pro spuštění dopředného řetězení a bude se pokoušet ověřovat všechny listové hypotézy a najít podle odpovědí uživatele všechny důsledky.

```
start_dopredu :-
    uklid,
    list List,
    write( '> Vybrany list je "' ),
    write( List ),
    write( '".' ), nl,
    dopredu( List ),
    write( '? Dalsi list? [a/n]: ' ),
    read( Odp ),
    Odp = n,
    write( 'Konec...' ),
    nl, !.
start_dopredu :-
    write( '> Nejsou dalsi listy.' ), nl, !.
```

Tím je celý inferenční mechanismus hotov a můžeme začít testovat. Opět se pokusíme ověřit tři živočichy, jako v předchozím příkladu. Začneme tentokrát lvem:

```
?- start_dopredu.
> Vybrany list je "saje_mleko".
> Hledame dusledek hypotezy: saje_mleko
? Je pravda, ze saje_mleko ? [a/n] a.
+ Z hypotezy saje_mleko plyne dusledek: savec
? Dalsi list? [a/n]: a.
> Vybrany list je "ma_peri".
> Hledame dusledek hypotezy: ma_peri
- Z hypotezy ma_peri nebylo nic odvozeno.
> Vybrany list je "leta".
> Hledame dusledek hypotezy: leta
- Z hypotezy leta nebylo nic odvozeno.
> Vybrany list je "ma_drapy".
> Hledame dusledek hypotezy: ma_drapy
? Je pravda, ze ma_drapy ? [a/n] a.
+ Z hypotezy ma_drapy plyne dusledek: selma
? Dalsi list? [a/n]: a.
> Vybrany list je "zere_maso".
> Hledame dusledek hypotezy: zere_maso
+ Z hypotezy zere_maso plyne dusledek: selma
? Dalsi list? [a/n]: a.
> Nejsou dalsi listy.

Yes
```

Z výpisu během činnosti inferenčního mechanismu vidíme tři řádky začínající znakem „+“. Jsou to zjištěné závěry: savec a šelma. Klasifikace lva je tedy správná.

Zkusíme, jak si systém poradí s druhým živočichem, tím je havran:

```
?- start_dopredu.
> Vybrany list je "saje_mleko".
> Hledame dusledek hypotezy: saje_mleko
? Je pravda, ze saje_mleko ? [a/n] n.
- Z hypotezy saje_mleko nebylo nic odvozeno.
> Vybrany list je "ma_peri".
> Hledame dusledek hypotezy: ma_peri
> Bude overovana hypoteza: savec
- Hypoteza savec neni splnena!
? Je pravda, ze ma_peri ? [a/n] a.
+ Z hypotezy ma_peri plyne dusledek: ptak
? Dalsi list? [a/n]: a.
> Vybrany list je "leta".
> Hledame dusledek hypotezy: leta
> Bude overovana hypoteza: savec
- Hypoteza savec neni splnena!
+ Z hypotezy leta plyne dusledek: ptak
? Dalsi list? [a/n]: a.
> Vybrany list je "ma_drapy".
> Hledame dusledek hypotezy: ma_drapy
> Bude overovana hypoteza: savec
- Hypoteza savec neni splnena!
? Je pravda, ze ma_drapy ? [a/n] n.
? Je pravda, ze zere_maso ? [a/n] a.
+ Z hypotezy ma_drapy plyne dusledek: dravec
? Dalsi list? [a/n]: a.
> Vybrany list je "zere_maso".
> Hledame dusledek hypotezy: zere_maso
> Bude overovana hypoteza: savec
- Hypoteza savec neni splnena!
+ Z hypotezy zere_maso plyne dusledek: dravec
? Dalsi list? [a/n]: a.
> Nejsou dalsi listy.

Yes
```

Havran byl správně klasifikován jako pták a dravec. Zbývá poslední ověření, a to je zajíc:

```
?- start_dopredu.
```

```
> Vybrany list je "saje_mleko".
> Hledame dusledek hypotezy: saje_mleko
? Je pravda, ze saje_mleko ? [a/n] a.
+ Z hypotezy saje_mleko plyne dusledek: savec
? Dalsi list? [a/n]: a.
> Vybrany list je "ma_peri".
> Hledame dusledek hypotezy: ma_peri
- Z hypotezy ma_peri nebylo nic odvozeno.
> Vybrany list je "leta".
> Hledame dusledek hypotezy: leta
- Z hypotezy leta nebylo nic odvozeno.
> Vybrany list je "ma_drapy".
> Hledame dusledek hypotezy: ma_drapy
? Je pravda, ze ma_drapy ? [a/n] n.
? Je pravda, ze zere_maso ? [a/n] n.
> Bude overovana hypoteza: ptak
- Hypoteza ptak neni splnena!
- Z hypotezy ma_drapy nebylo nic odvozeno.
> Vybrany list je "zere_maso".
> Hledame dusledek hypotezy: zere_maso
> Bude overovana hypoteza: ptak
- Hypoteza ptak neni splnena!
- Z hypotezy zere_maso nebylo nic odvozeno.
> Nejsou dalsi listy.

Yes
```

Z výpisu plyne, že zajíc je savec, což je v pořádku. Ale naše báze znalostí umí klasifikovat i býložravce. K tomuto závěru jsme pomocí dopředného řetězení nedošli. Kde je chyba?

Pozorného čtenáře možná napadne, zda nejde o podobnou chybu, jaká byla u zpětného řetězení. Ano, je tomu tak. I zde predikát „krok_dopredu“ řeší jen jediný krok od předpokladu k důsledku. Chybí tedy skutečné **řetězení**. Doplnění rekurze je velmi snadné a nebudeme jej zde uvádět. Každý si v rámci seznamování a testování inferenčního mechanismu úpravu provede sám.

2 Cvičení

Expertním systémům jsou pro jejich náročnost věnována čtyři cvičení.

2.1 Seznámení a testování inferenčních mechanismů

Úkolem cvičení je testování navržených inferenčních mechanismů. Předpokládá se také rozšíření báze znalostí o 5 až 10 pravidel a testování dosažitelnosti všech cílů. V rámci cvičení by se každý měl pokusit správně doplnit mechanismus dopředného řetězení o chybějící rekurzi.

2.2 Výběr tématu pro návrh báze znalostí

Cvičení bude věnováno rozboru možných témat, vhodných pro návrh vlastní báze znalostí. Je zapotřebí zvážit, jaký formát báze znalostí je pro dané téma vhodný, zda lze skutečně sestavit pravidla pro odvozování a případné zhodnocení, zda vybrané téma není příliš náročné, nebo naopak, příliš jednoduché.

Odvozovací pravidla je vhodné znázornit graficky. Celý problém se tak dá lépe analyzovat.

2.3 Tvorba vlastní báze znalostí

Ve cvičení si každý zformuluje odvozovací pravidla pro svou bázi znalostí. Provedou se první testy spolupráce mezi bází znalostí a inferenčním mechanismem.

2.4 Ladění a předvedení

Čtvrté cvičení bude věnováno závěrečnému testování a prezentaci vlastního experního systému.