



Technická univerzita v Liberci
Ekonomická fakulta

Uměla inteligence

Semestrální práce

Vlk, koza a zelí v Prologu

Adam Řehořek

Manažerská informatika

Magisterské: 1. ročník

1 Úvod

Pro svou semestrální práci jsem si vybral práci s jazykem Prolog. V průběhu cvičení z předmětu *Umělá inteligence* mě tento jazyk zaujal a doma jsem si vyzkoušel několik jednodušších programů. Většina z nich ale neměla nic společného s umělou inteligencí. Nakonec jsem si zvolil všem známou úlohu *Koza, zelí a vlk*. Tu jsem vypracoval v programu *SWI-Prolog*, což je volně šiřitelný překladač jazyka Prolog.

2 Popis úlohy

Koza, zelí, vlk je logická hádanka, v níž hráč převozník musí přemístit kozu, zelí a vlka z jednoho břehu řeky na druhý. Ve hře platí následující pravidla:

- Hráč má k dispozici loďku, do níž se vejde převozník a maximálně jeden předmět.
- Loďku musí vždy řídit převozník.
- Pokud zůstanou spolu na jednom břehu bez dozoru převozníka koza a vlk, tak vlk sežere kozu.
- Pokud zůstane spolu na jednom břehu bez dozoru převozníka koza a zelí, tak koza sežere zelí.

Tuto úlohu jistě všichni dobře známe a již jsme ji v nějakém jazyku programovali. Jak popisuje stránka www.algoritmy.net: „Z algoritnického hlediska se jedná o prohledávání stavového prostoru s návratem (backtracking). Backtrackingový algoritmus v každém svém kroku zkusí převézt jednu věc na druhý břeh a zkontroluje, jestli nedošlo k porušení podmínek úlohy (tj. nebyla sežrána koza nebo zelí). Pokud kontrola proběhne úspěšně, tak se proces opakuje. Pokud daný převoz způsobí nekonzistenci úlohy, tak se algoritmus vrátí do místa posledního rozhodnutí a převezde jinou věc (nebo se vrátí o rozhodnutí výše, pokud již byly všechny možnosti vyzkoušeny). Organizovaným prohledáváním ve stylu pokus-omyl algoritmus nalezne hledanou sekvenci kroků (řešení).“.

Tudíž šlo jen o to, zjistit, jak to nejprůhledněji napsat v jazyce Prolog.

3 Řešení úlohy v Prologu

Pro vyřešení této úlohy v Prologu je potřeba zakódovat 4 objekty (převozník, vlk, koza, zeli) jako list. Pak *v* značí východní břeh a *z* břeh západní.

Stav *[z,z,z,z]* pak tedy znamená, že všichni jsou na západním břehu. Toto je také počáteční stav, konečný stav je tedy *[v,v,v,v]*.

Při každém *pohybu* může převozník vzít s sebou vlka, kozu, zeli nebo nic. Každý pohyb může být reprezentován odpovídajícím atomu (v jazyku Prolog značí jednoduchou datovou položku): *vlk,koza,zeli,nic*. Každý *pohyb* mění jedno rozestavení v jiné. To může být v Prologu zapsáno takto: *move(Rozestaveni,pohyb,NoveRozestaveni)*, kde *Rozestaveni* je momentální postavení objektů na březích, *pohyb* je jeden ze čtyř výše uvedených pohybů a *DalšíRozestaveni* značí výsledek aplikování *pohybu* na původní rozestavení. Např. *pohyb([z,z,z,z],vlk,[v,v,z,z])*.

Jelikož tu však může být spousta možností k zakódování, je lepší danou skutečnost zobecnit. A to na *move([w,w,koza,zeli],vlk,[e,e,koza,zeli])*., kde jsou *koza* a *zeli* proměnné. K výše uvedeným skutečnostem, je možné příkaz ještě více zobecnit a to na *pohyb([X,X,koza,zeli],vlk,[Y,Y,koza,zeli]) :- zmena(X,Y)*., který předpokládá, že prediktát *zmena* je definován jako: *zmena(e,w)*. a *zmena(w,e)*.

Všechny možné pohyby jsou tedy zakódovány následovně:

zmena(e,w).

zmena(w,e).

pohyb([X,X,koza,zeli],vlk,[Y,Y,koza,zeli]) :- zmena(X,Y).

pohyb([X,vlk,X,zeli],koza,[Y,vlk,Y,zeli]) :- zmena(X,Y).

pohyb([X,vlk,koza,X],zeli,[Y,vlk,koza,Y]) :- zmena(X,Y).

pohyb([X,vlk,koza,zeli],nic,[Y,vlk,koza,zeli]) :- zmena(X,Y).

Dále bylo potřeba, aby program testoval bezpečnost převozu, aby nebylo nic sežráno. Proto si definujeme prediktát *oneEq(X,Y,Z)*, který bude pravda když minimálně jeden z *Y* nebo *Z* bude roven *X*.

alespon(X,X,_). a *alespon(X,_,X).*

Myšlenkou je, aby byl minimálně jeden z dvojice koza a vlk na stejném břehu jako převozník a také, aby minimálně jeden z dvojice koza a zeli byl na stejném břehu jako převozník. To lze zakódovat takto:

*bezpecne([prevoznik,vlk,koza,zeli]) :- alespon(prevoznik,koza,vlk),
alespon(prevoznik,koza,zeli).*

Když máme vyřešené možnosti pohybu i to, kdy je situace na březích bezpečná, můžeme to dát dohromady a tuto úlohu vyřešit. *Reseni* je definováno startovacím rozmístěním a listem pohybů, které nás dostanou do cílového rozmístění. Pokud budeme chtít vyřešit *reseni([e,e,e,e],[])*, řešením bude prázdný list, jelikož nebudou potřeba žádné převozy mezi břehy. V opačném případě je *reseni* definováno rekurzivně, jako jeden tah, který vás vezme do bezpečného rozmístění následováno dalším *reseni*. Tato rekurze je zakódována takto:

```
reseni([v,v,v,v],[ ]).
reseni(Rozestaveni,[PrvniTah/OstatniTahy]) :-
    pohyb(Rozestaveni,PrvniTah,DalsiRozestaveni),
    bezpecne(DalsiRozestaveni),
    reseni(DalsiRozestaveni,OstatniTahy).
```

Jedno varování na závěr. V Prologu není definováno, jak dlouhé musí řešení být. Pokud bychom neomezili počet kroků, program by se dostal do smyčky, například s nekonečně mnoho převozů ničeho tam a zpátky. Jelikož víme, jak je dlouhé nejkratší možné řešení, rovnou tuto délku dáme do dotazu.

```
?- length(X,7), reseni([z,z,z,z],X).
```

Tento dotaz však zobrazí několikrát stejné řešení. Proto použijeme dotaz, který odstraní všechna duplikovaná řešení.

```
?- length(X,7), setof(t,reseni([z,z,z,z],X),_).
```

Po použití tohoto dotazu nám program vyhodí dva výsledky.

```
X = [koza, nic, vlk, koza, zeli, nic, koza] ;
```

```
X = [koza, nic, zeli, koza, vlk, nic, koza].
```

První výsledek znamená, že první převozník poveze kozu, zpátky pojede s ničím, na východní břeh poté vezme vlka a zpátky kozu, na východní břeh poté zeli, vrátí se prázdný a nakonec tam odveze kozu.

Druhý výsledek znamená, že nejdříve převozník poveze kozu, zpátky pojede s ničím, poté převezme zeli a zpátky kozu, dále poveze na druhý břeh vlka a zpátky pojede prázdný a nakonec převezme na druhý břeh kozu.

3.1 Zdrojový kód a výsledek

```
zmena(v,z).
zmena(z,v).
pohyb([X,X,Koza,Zeli],vlk,[Y,Y,Koza,Zeli]) :- zmena(X,Y).
pohyb([X,Vlk,X,Zeli],koza,[Y,Vlk,Y,Zeli]) :- zmena(X,Y).
pohyb([X,Vlk,Koza,X],zeli,[Y,Vlk,Koza,Y]) :- zmena(X,Y).
pohyb([X,Vlk,Koza,Zeli],nic,[Y,Vlk,Koza,Zeli]) :- zmena(X,Y).

alespon(X,X,_).
alespon(X,_,X).

bezpecne([Prevoznik,Vlk,Koza,Zeli]) :-
    alespon(Prevoznik,Koza,Vlk),
    alespon(Prevoznik,Koza,Zeli).

reseni([v,v,v,v],[ ]).
reseni(Rozestaveni,[PrvniTah|OstatniTahy]) :-
    pohyb(Rozestaveni,PrvniTah,DalsiRozestaveni),
    bezpecne(DalsiRozestaveni),
    reseni(DalsiRozestaveni,OstatniTahy).

14 ?- length(X,7), setof(t,reseni([z,z,z,z],X),_).
X = [koza, nic, vlk, koza, zeli, nic, koza] ;
X = [koza, nic, zeli, koza, vlk, nic, koza].
```

4 Závěr

Jsem rád, že jsem se na hodinách seznámil s jazykem Prolog a mohl jsem si v něm zkusit něco naprogramovat. Je jasné, že není tak rozšířený a využíváný, jako jiné programovací jazyky, ale určitě to byla zajímavá zkušenost.