Neprocedurální programování

Dokumentace zápočtového programu
v jazyce Prolog

Zebra Solver

Autor:
Petr Martišek

Cvičící:
RNDr. Rudolf Kryl

13. června 2013

1 Anotace

Program Zebra Solver je aplikace sloužící k automatickému řešení tzv. Einsteinových hádanek, v angličtině rovněž „grid puzzles“, často také označované slovem „zebra“. Tato přezdívka vznikla na základě originálního zadání uveřejněného v časopise Life International roku 1962. Autorství je připisováno právě Albertu Einsteinovi.

Naše aplikace umí vyřešit nejen tuto originální, ale de facto jakoukoliv hádanku tohoto typu, která splňuje jisté nároky na tvar podmínek (specifikováno níže). Tyto nároky jsou založeny na původním zadání a zobecněny tak, aby bylo možno řešit co největší spektrum hádanek.

2 Přesné zadání

Zobecněné znění problému, který program řeší, je následující: Existuje N objektů. Tyto objekty jsou umístěny tak, že je lze jednoznačně očíslovat $1..N$ (např. domy stojící v řadě). Všechny objekty mají stejnou sadu M vlastností. Každá vlastnost může nabývat celkem N různých hodnot. Každý objekt nabývá u každé vlastnosti právě jedné hodnoty a žádné dva objekty nenabývají u žádné vlastnosti téže hodnoty.

Pozn.: Mezi vlastnosti je automaticky přidána vlastnost index*, která udává pořadí objektu v posloupnosti. Tuto vlastnost může uživatel používat při zadávání podmínek, aniž by ji musel explicitně specifikovat.*

Dále je zadáno libovolné množství podmínek. Podmínky dávají omezení na distribuci hodnot jednotlivých vlastností a na pořadí objektů. Rozlišíme dva základní typy podmínek:

1. **unární podmínky**
   Tedy podmínky mluvící o jednom objektu. Formálně se jedná o výčet dvojic *Vlastnost - Hodnota*, které musí u nějakého objektu v prostoru řešení platit současně. Počet dvojic může být libovolné číslo mezi 1 a M.

   Příklady:
   „Angličan bydlí v červeném domě.“ → *(Národnost – Angličan, Barva – Červená)*
   „Ten, kdo kouří Marlboro, bydlí v modrém domě a chová kočku.“ → *(Cigarety – Marlboro, Barva – Modrá, Zvíře – Kočka)*
   „Někdo chová koně.“ → *(Zvíře – Kůň)*

2. **relační podmínky**
   Podmínky mluvící o vztahu několika objektů. Formálně jsou to dva výčty: První je výčet dvojic *Vlastnost - Hodnota*, z nichž každá dvojice jednoznačně identifikuje nějaký objekt. Počet objektů, jejichž vztahy specifikujeme, je číslo mezi 1 a N (nicméně délka prvního seznamu může teoreticky být až $N \times M$, protože některé dvojice mohou specifikovat týž objekt).

   Druhým argumentem této podmínky je výčet relací. Relace je libovolný k-ární predikát, kde k je počet objektů v prvním výčtu, který bere jako argumenty k objektů, kdy objekt je dán jako *seznam hodnot, kterých nabývají jeho vlastnosti* (tedy seznam délky M).

   Příklady:
   „Francouz bydlí hned vpravo vedle chovatele psa.“ → *((Národnost – Francouz, Zvíře – Pes), (JeSoused, Vpravo))*
   „Modrý dům stojí na kraji řady.“ → *((Barva – Modrá), (NaKraji))*
   „Chovatel šneků bydlí mezi Němcem a kuřákem Chesterfieldek.“ → *((Zvíře – Šneci, Národnost – Němec, Cigarety – Chesterfield), (BydlíMezi))*

   kde *NaKraji* je unární, *Soused* a *Vpravo* jsou binární a *BydlíMezi* je ternární predikát.

Na závěr můžeme položit libovolné množství otázek na konkrétní hodnotu nějaké vlastnosti nějakého objektu. Otázka je formálně dvojice *((Vlastnost – Hodnota) – Vlastnost)*, kde první položka, dvojice *(Vlastnost – Hodnota)*, určuje konkrétní objekt a druhá položka udává vlastnost, na jejíž hodnotu se u tohoto objektu ptáme.
Příklad:
„Jakou barvu má dům, v němž bydlí Němec?“ → *((Národnost – Němec), Barva)*.

3 Formát vstupu

Program je spouštěn predikátem `zebra(+PocetObjektu, +Vlastnosti, +Podminky, +Otazky, -Odpovedi)`. Formát jednotlivých argumentů je následující:

1. **PocetObjektu**
   Libovolné přirozené číslo, udávající počet objektů.

2. **Vlastnosti**
   Seznam vlastností objektů, jiné vlastnosti než zde uvedené nebudou v podmínkách rozpoznány.
   Např.: `[barva, narodnost, zvire, cigarety, napoj]`

3. **Podminky**
   Seznam podmínek v jednom z následujících formátů:

   (a) **unární podmínka**

   ```
   podminka([(Vlastnost1, Hodnota1),(Vlastnost2, Hodnota2), ...]),
   ```

   kde `Vlastnost1, Vlastnost2,...` jsou některé z vlastností uvedených v druhém argumentu predikátu `zebra` a `Hodnota1, Hodnota2,...` jsou hodnoty, jichž mají tyto vlastnosti u nějakého objektu nabývat současně. Délka seznamu je celé číslo z intervalu $1..M$. Pokud se v seznamu pokusíme definovat pro tutéž vlastnost více různých hodnot, je to považováno za chybu a program zfailuje. Pokud se v seznamu vyskytne vlastnost nedefinovaná v druhém argumentu predikátu `zebra`, program zfailuje. Výjimkou je zabudovaná vlastnost `index`.

   (b) **relační podmínka**

   ```
   podminka([(Vlastnost1, Hodnota1),(Vlastnost2, Hodnota2), ...],
   [Relace1, Relace2, ...]),
   ```

   kde dvojice `(Vlastnost, Hodnota)` určují konkrétní objekty, pro něž musí platit všechny relace dané seznamem v druhém argumentu. Tento typ podmínek lze číst jako: *Pro objekt, jehož `Vlastnost1` nabývá hodnoty `Hodnota1`, a objekt, jehož `Vlastnost2` nabývá hodnoty `Hodnota2`, (a objekt . . .), platí vztah daný relací `Relace1` a současně vztah daný relací `Relace2` (a současně . . .).*
   Arita použitých relací musí odpovídat délce seznamu v prvním argumentu. Předdefinované relace jsou `soused/2, vpravo/2` a `vlevo/2.` Další relace je třeba uživatelsky definovat. Každá relace musí přijímat argumenty ve tvaru `[Hodnota0, Hodnota1, Hodnota2, Hodnota3, ..., HodnotaM]`, kde M je celkový počet vlastností, `Hodnota0` odpovídá vlastnosti `index` a jednotlivé hodnoty $1..M$ odpovídají vlastnostem v pořadí, v jakém jsou definovány v druhém argumentu predikátu `zebra`. Jednotlivé položky seznamu jsou buď konkrétní hodnoty nebo volné proměnné. Tato struktura je vlastně interní reprezentací konkrétního objektu.

4. **Otazky**
   Seznam otázek ve formátu:

   ```
   otazka((Vlastnost1, Hodnota1), Vlastnost2)
   ```

   kde `Vlastnost1` a `Hodnota1` určují konkrétní objekt a `Vlastnost2` udává vlastnost, jejíž hodnota nás u tohoto objektu zajímá.

5. **Odpovedi**
   Seznam hodnot, které jsou odpověďmi na otázky z předešlého argumentu. Délka tohoto seznamu musí být stejná, jako počet otázek. Během výpočtu se tento seznam naplní konkrétními hodnotami.

3.1 Příklad zadání

Ukažme si správný formát vstupu na příkladu. Následuje zadání originální originální hádanky (viz. např. `http://en.wikipedia.org/wiki/Zebra_Puzzle`) přepsáno do správného formátu:

```
zebra(
    5,
    [color, pet, beverage, nationality, cigarets],
    [
    podminka([(nationality, englishman),(color,red)]),
    podminka([(nationality,spaniard), (pet,dog)]),
    podminka([(color, green),(beverage, coffee)]),
    podminka([(nationality,ukrainian),(beverage, tea)]),
    podminka([(color, green),(color,ivory)], [soused,vpravo]),
    podminka([(cigarets,old_gold),(pet, snails)]),
    podminka([(color,yellow),(cigarets, kools)]),
    podminka([(beverage, milk),(index, 3)]),
    podminka([(nationality, norwegian),(index,1)]),
    podminka([(cigarets,chesterfield),(pet,fox)], [soused]),
    podminka([(cigarets,kools),(pet,horse)],[soused]),
    podminka([(cigarets,lucky_strike),(beverage, orange_juice)]),
    podminka([(nationality,japanese),(cigarets, parliaments)]),
    podminka([(nationality,norwegian),(color,blue)],[soused])
    ],
    [
    otazka((pet,zebra), color),
    otazka((beverage, water), cigarets)
    ],
    [O1,O2]
    ).
```

Po skončení programu se v proměnné `O1` bude nacházet barva domu, ve kterém je chována zebra, a v proměnné `O2` značka cigaret, jež kouří ten, kdo pije vodu.

4 Algoritmus

V prvním kroku vyextrahujeme z podmínek a z otázek jednotlivé částečné objekty. Připomeňme, že objekt je reprezentován jako seznam hodnot, přičemž částečným objektem nazveme takový objekt, jehož některé hodnoty jsou volné proměnné. Např. z podmínky `podminka([(nationality,`

`englishman), (color,red)])` získáme částečný objekt `[red, _, _, englishman, _]`, z podmínky `podminka([(nationality,norwegian), (color,blue)], [soused])` získáme dva částečné objekty `[_, _, _, norwegian, _]` a `[blue, _, _, _, _]`. Z otázky `otazka((pet, zebra), color)` získáme částečný objekt `[_, zebra, _, _, _]`. Současně upravujeme seznam podmínek a odstraňujeme všechny unární podmínky, neboť nám již k ničemu nebudou. Relační podmínky ponecháme, protože budeme potřebovat kontrolovat platnost všech příslušných relací.

V druhém kroku sloučíme ty částečné objekty, jež se shodují v nějaké vlastnosti. Např. objekty `[blue, _, tea, _, _]` a `[blue, _, _, englishman, _]` sloučíme do jediného částečného objektu `[blue, _, tea, englishman, _]`, protože se shodují ve vlastnosti `color`. Tato operace jednak zrychlí celý algoritmus, jednak zaručí, že ve výsledku nevzniknou dva různé objekty shodující se v nějaké vlastnosti a jednak odhalí nekorektní zadání, kdy by jednomu objektu bylo přiřazeno více hodnot téže vlastnosti. Příkladem takového chybného zadání je např. seznam podmínek ve tvaru:

```
[
...,
podminka([(nationality, englishman) , (color, red)]),
...,
podminka([(nationality, englishman), (color,blue)]),
...
].
```

Podle hodnoty vlastnosti `nationality` by se zjevně mělo jednat o týž objekt, ale vlastnost `color` by pak musela nabývat dvou různých hodnot. Pro takovéto zadání program nehledá řešení a rovnou zfailuje.

Ve třetím kroku si vytvoříme prázdnou tabulku $N \times M$, kde N je výsledný počet objektů (daný prvním argumentem predikátu `zebra`) a M počet vlastností. Do této tabulky budeme postupně dosazovat částečné objekty z redukovaného seznamu získaného v druhém kroku. Zde využíváme dvou základních stavebních kamenů Prologu – unifikace a backtrackingu. Ze seznamu částečných objektů vezmeme první v pořadí, odstraníme jej ze seznamu a pokusíme se jej unifikovat s některým řádkem tabulky. Objekty jsou seznamy, tudíž je lze unifikovat tehdy, pokud lze unifikovat všechny odpovídající prvky těchto seznamů. Těmito prvky jsou hodnoty jednotlivých vlastností a ty lze unifikovat tehdy, pokud je alespoň jedna z hodnot volná proměnná nebo pokud jsou hodnoty shodné. Pokud se nám unifikace podaří, ověříme platnost podmínek a pokračujeme dalším objektem ze seznamu. Pokud se unifikace či ověření podmínek nepodaří, backtrackujeme. Tedy zkusíme nalézt jiné umístění pro předchozí umístěný objekt, případně objekt před ním atd.

Ověření podmínek probíhá při každém přidání nějakého částečného objektu do tabulky. Postupně procházíme seznam podmínek, pro každou jednotlivou podmínku najdeme v tabulce objekty, jichž se týká (pokud už v tabulce jsou), a zavoláme na ně příslušné predikáty pomocí procedury *apply/2*.

Program nalezne postupně všechna možná řešení (pokud si je uživatel vyžádá stiskem středníku). Procházíme tedy celý stavový strom, ořezaný podmínkami. Stavový strom je N-ární a má hloubku P, kde P je délka redukovaného seznamu částečných objektů získaných z podmínek a otázek. Tedy pro každý částečný objekt ze seznamu máme na výběr N řádků tabulky, kam jej můžeme zkusit umístit, pro další objekt opět N možností atd. pro všechny částečné objekty. Počet stavů, do kterých se dostaneme, lze tedy shora odhadnout jako N^P. V každém stavu proběhne pokus o unifikaci v čase $O(M)$ a ověření podmínek v čase $O(Q \times A \times N \times M)$, kde Q je počet relačních podmínek, A je horní odhad arity relací, N reprezentuje hledání objektu v tabulce a $O(M)$ je odhad časové složitosti ověření jedné podmínky (u objektů nalezneme vlastnost, o níž relace mluví, a pak v konstatním čase provedeme porovnání). Celkový počet operací lze tedy odhadnout jako

$$O(N^P \times (M + Q \times A \times N \times M))$$

Exponenciální složitost není, vzhledem k použití backtackingu, nijak překvapivá. Přesto však zkusme porovnat tento algoritmus s naivní implementací, kdy bychom postupně zkoušeli všechny možné konfigurace zaplnění tabulky a pokaždé kontrolovali platnost všech podmínek. Vychází nám odhad $O((N!)^M \times P \times A \times N \times M)$.

5 Program

Následuje podrobný popis jednotlivých částí programu (resp. těch, které si podrobný komentář zaslouží).

1. **Přidání vlastnosti index** Trivální procedura `pridejIndex` přidá před zadaný seznam vlastností vlastnost *index*.

2. **Extrakce částečných objektů z podmínek a otázek**
   Hlavní procedury jsou `extrahujHodnotyZPodminek`, resp. `extrahujHodnotyZOtazek`. První zmíněná má dvě klauzule, jednu pro ošetření unárních pomínek, které ze senamu mažeme, a druhou pro relační podmínky, které ponecháme. Obě klauzule prochází seznam podmínek a na každou zavolají proceduru `extrahuj`. Ta má opět dvě verze. Pro unární podmínky jednoduše zavoláme proceduru `vytvorObjekt`, které předáme seznam dvojic `(Vlastnost, Hodnota)`. Pro relační potřebujeme vytvořit objektů více, procházíme tedy rekurzivně seznam v prvním argumentu podmínky a pro každou dvojici vytváříme nový objekt. Procedura `extrahuj` vrací pro každou podmínku seznam částečných objektů z ní vytvořených.

   Procedura `vytvorObjekt` nejprve vytvoří prázdný objekt (= seznam volných proměnných délky M) a pak do něj v rekurzivní proceduře `vObjekt` dosazuje jednotlivé hodnoty pomocí predikátu `pridejHodnotu`.

   Procedura `extrahujHodnotyZOtazek` je ještě jednodušší. Pro každou otázku vytváříme pouze jeden objekt, a to z dvojice `(Vlastnost, Hodnota)`, která tvoří první argument predikátu `otazka`. Stačí nám tedy přímočaře zavolat proceduru `vytvorObjekt`.

3. **Sloučení částečných objektů shodujících se v nějaké vlastnosti**
   Hlavní procedurou je zde `zahusti`. Ta bere jako první argument seznam částečných objektů získaný v předchozím kroku a vrací jako druhý argument rovněž seznam částečných objektů, v němž jsou však již všechny objekty, jež se shodují v nějaké vlastnosti, sloučeny do jednoho. Procedura postupně bere jednotlivé objekty seznamu a snaží se je sloučit s některým ze zbytku seznamu. K tomu slouží procedura `zaclen`. Ta má 4 argumenty: začleňovaný objekt; seznam, do nějž začleňujeme; upravený seznam bez objektů, které se povedlo sloučit se začleňovaným; výsledný sloučený objekt. Procedura při korektním zadání uspěje vždy, tedy i pokud nelze začleňovaný objekt s ničím sloučit. Pak je jednoduše čtrvtý argument roven prvnímu (a třetí druhému). Jediný případ, kdy dojde k selhání, je v případě chybného zadání, kdy by jednomu objektu bylo přiděleno více hodnot téže vlastnosti.

   Procedura `zaclen` prochází seznam zbylých objektů a u každého zkouší, zda je možno jej sloučit se začleňovaným. K tomu slouží procedura `shodaVHodnote` **následovaná řezem**. Uspěje-li `zaclen`, nahradí začleňovaný objekt tímto novým sloučeným a pokračuje v procházení seznamu (toto je důležité pro případ, kdy v seznamu je hned několik objektů slučitelných do jednoho). Neshodují-li se řádky v nějaké hodnotě, pokračujeme dalším řádkem.

   Slučování probíhá následovně: `shodaVHodnote` projde oba řádky a pokusí se najít nějakou vlastnost, v níž objekty nabývají stejné hodnoty. Neuspěje-li, řádky nejsou kompatibilní a pokračujeme dál. Avšak v případě, že uspěje, znamená to nejen, že řádky lze sloučit,

ale dokonce že řádky **musí jít** sloučit. Proto následuje operátor řezu a až potom operátor =, který se pokusí řádky unifikovat. V případě korektního zadání uspěje. Neúspěch totiž znamená, že některou z vlastností nelze unifikovat, což nastane tehdy, pokud oba objekty nabývají pro tuto vlastnost konkrétních a navzájem různých hodnot. Což indikuje chybu v zadání, neboť díky proceduře `shodaVHodnotě` již víme, že objekty se v nějaké hodnotě shodují.

4. **Vytvoření prázdné tabulky** Tato část programu snad nevyžaduje příliš podrobný komentář. Za zmínku snad stojí jen to, že první sloupec tabulky (vlastnost `index`) je při konstrukci rovnou vyplněn čísly $1..N$.

5. **Zaplnění tabulky**
   Jádro programu. Hlavní procedurou je `zaplnTabulku`. Ta vezme první ze seznamu částečných objektů, pokusí se jej vložit do tabulky procedurou `vlozObjekt`, zkontroluje platnost relačních podmínek procedurou `zkontroluj` a v případě úspěchu rekurzivně zavolá sama sebe na další částečný objekt.

   Procedura `vlozObjekt` prochází tabulku řádek po řádku a snaží se najít takový, který by šel unifikovat z objektem, který se snaží začlenit. Nelze-li unifikovat, rekurzivně procházíme zbytek tabulky.

   Procedura `zkontroluj` rekurzivně prochází seznam podmínek a pro každou nejprve najde v tabulce řádky, jichž se podmínka týká, a následně zkontroluje platnost dané podmínky predikátem `plati`. Nepodaří-li se odpovídající řádky v tabulce najít, znamená to, že ještě v tabulce nejsou a tudíž podmínka rozhodně nebude porušena. V takovém případě nic neověřujeme a vezmeme další podmínku.

   Procedura `plati` aplikuje na seznam objektů postupně všechny relace dané druhým argumentem relační podmínky. K tomu poslouží zabudovaná procedura `apply`.

6. **Nalezení odpovědí na otázky v tabulce**
   Procedura `zodpovez` prochází seznam otázek a na každou zavolá proceduru `zodp`. Ta nalezne v již vyplněné tabulce řádek odpovídající dvojici `(Vlastnost, Hodnota)` dané prvním argumentem predikátu `otazka` (predikát `najdiRadek`) a následně na tomto řádku nalezne hodnotu odpovídající vlastnosti dané druhým argumentem predikátu `otazka` (predikát `najdiHodnotu`).

7. **Výpis řešení**
   Sada jednoduchých procedur vypíše na výstup vyplněnou tabulku a odpoví celými větami na zadané otázky. Tabulka může být neúplná (některé položky mohou být stále volné proměnné), což nám ovšem ani v nejmenším nebrání v nalezení řešení. Stisknutím středníku může uživatel odmítnout řešení a vyzvat program k nalezení dalšího. Za různá řešení považujeme ta, jejichž výsledné tabulky se liší. To znamená, že dvě různá řešení mohou mít stejné odpovědi na otázky (ale různé tabulky, tj. může se lišit např. pořadí objektů).

6 Testovací data

Součástí řešení jsou i vhodná testovací data umístěná v souboru `testovaci_data.pl`. Volání probíhá pomocí unární procedury `testuj`, která bere za argument název konkrétního testovacího příkladu. Např. příkaz `testuj(original)` zavolá proceduru `zebra` s argumenty odpovídajícími orignálnímu zadání Einsteinovy hádanky. Testovací příklady nijak nevyužívají vyplněný seznam odpovědí v posledním argumentu, odpovědi jsou pouze vypsány pomocí naší definované procedury pro výpis.

Poskytnuté testovací příklady jsou následující:

6.1 original

Již zmíněné originální zadání. Zní takto:

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets. One other thing: in statement 6, right means your right.

Pro toto zadání existuje právě jedno řešení. Výsledná tabulka je kompletní (bez anonymních proměnných).

6.2 priklad1

Zde pouze ukazujeme korektnost zahušťování seznamu částečných objektů. Pro názornost je tento příklad ošetřen upravenou variantou hlavní procedury, a to s názvem `zebraSVypisem`. Jediná odlišnost je v tom, že v průběhu algoritmu dojde k vypsání seznamu objektů před a po zahuštění.

Z podmínek se extrahují hned 4 objekty, shodující se ve vlastnosti `barva`. Na výpisech lze názorně vidět, že správně dojde ke sloučení všech 4 do jediného objektu.

6.3 priklad2

Rovněž příklad demonstrativního charakteru. Jedná se o originální hádanku s přidanou relační podmínkou, která udává vztah 12 objektů. Je tedy vidět, že některé dvojice `(Vlastnost, Hodnota)` mohou specifikovat týž objekt. Použili jsme vlastní proceduru `notFirst`, tentokrát s dvanácti argumenty, která uspěje, pokud ani jeden z 12 objektů daných argumenty není na první pozici. Díky vhodně zvoleným argumentům je výsledek příkladu stejný jako výsledek originální hádanky.

6.4 chyba1

Ukázka chybného zadání. Jedné vlastnosti téhož objektu (určeného hodnotou `zluta` vlastnosti `barva`) se snažíme v různých podmínkách přiřadit dvě různé hodnoty (v tomto konkrétním případě dvě různé značky cigaret).

6.5 chyba2

Další příklad chyby v zadání. Jedná se de facto o tutéž chybu jako předchozím případě, tentokrát však k pokusu o přiřazení různých hodnot téže vlastnosti dochází v rámci jediné unární podmínky.

6.6 books

Variace originální hádanky. Viz `http://brainden.com/einsteins-riddles.htm`. Zadání zní:

> Another example of hard grid puzzles (just like Einstein's) was published in the QUIZ 11/1986. Eight married couples meet to lend one another some books. Couples have the same surname, employment and car. Each couple has a favorite color. Furthermore we know the following facts:
>
> 1. Daniella Black and her husband work as Shop-Assistants.
> 2. The book "The Seadog"was brought by a couple who drive a Fiat and love the color red.
> 3. Owen and his wife Victoria like the color brown.
> 4. Stan Horricks and his wife Hannah like the color white.
> 5. Jenny Smith and her husband work as Warehouse Managers and they drive a Wartburg.
> 6. Monica and her husband Alexander borrowed the book "Grandfather Joseph".
> 7. Mathew and his wife like the color pink and brought the book "Mulatka Gabriela".
> 8. Irene and her husband Oto work as Accountants.
> 9. The book "We Were Five"was borrowed by a couple driving a Trabant.
> 10. The Cermaks are both Ticket-Collectors who brought the book "Shed Stoat".
> 11. Mr and Mrs Kuril are both Doctors who borrowed the book "Slovacko Judge".
> 12. Paul and his wife like the color green.
> 13. Veronica Dvorak and her husband like the color blue.
> 14. Rick and his wife brought the book "Slovacko Judge"and they drive a Ziguli.
> 15. One couple brought the book "Dame Commissar"and borrowed the book "Mulatka Gabriela".
> 16. The couple who drive a Dacia, love the color violet.
> 17. The couple who work as Teachers borrowed the book "Dame Commissar".
> 18. The couple who work as Agriculturalists drive a Moskvic.
> 19. Pamela and her husband drive a Renault and brought the book "Grandfather Joseph".
> 20. Pamela and her husband borrowed the book that Mr and Mrs Zajac brought.
> 21. Robert and his wife like the color yellow and borrowed the book "The Modern Comedy".
> 22. Mr and Mrs Swain work as Shoppers.
> 23. "The Modern Comedy"was brought by a couple driving a Skoda.
>
> Is it a problem to find out everything about everyone from this information?

Více objektů i jejich vlastností, podmínky jsou až na jedinou výjimku unární, s různým počtem dvojic `(Vlastnost, Hodnota)`. Hádanka rovněž neobsahuje žádnou otázku, výsledkem má být pouze kompletní tabulka. Zajímavostí je, že dvě vlastnosti (`brought` a `borrowed`) nabývají stejných hodnot. Toho lze využít pro zkompletování tabulky (která by pouze na základě podmínek obsahovala i volná místa), a to dodefinováním unárních podmínek s pouze jedinou

vlastností, např. `podminka([(brought,the_seadog)])`. Takto vlastně pouze předáme programu informaci o tom, že jedna z hodnot, kterých vlastnost `brought` nabývá, je i `the_seadog`. Definujeme takové podmínky pro všechny knihy a pro obě vlastnosti `brought` a `borrowed`.

Příklad rovněž ukazuje použití uživatelsky definované relace `borrow` (je definována v témže souboru).

Příklad je rozsahem větší a na výpočtu je to znát – nalezení první odpovědi trvá několik vteřin. Poněkud nepříjemnou vlastností je to, že není požadováno specifické pořadí objektů. Náš program ovšem s pořadím pracuje a různá pořadí těch samých objektů považuje za různá řešení. Po stisku středníku bude tedy náš program zobrazovat postupně všechny permutace řádků téže tabulky.

6.7 heroes

Viz `http://answers.yahoo.com/question/index?qid=20111011084400AAjkXDH`. Tento příklad ukazuje použití hned několika uživatelsky definovaných relací. Nejčastěji je to způsobeno použitím negace v zadání, např „The lance is **not** carried by the snake.“ Tuto podmínku bohužel vzhledem ke způsobu výpočtu nelze považovat za unární a je nutno pomoct si speciálním predikátem.

Who are the heroes, what order do they stand, what armor do they wear, what sigil do they bear, how tall are they, what is their weapon of choice, and who bears the golden sigil? NOTE: The heroes vary in height by 1 foot increments, 3 feet being the shortest hero.

1. The lion stands in the middle
2. The bearer of the white sigil is not the tallest or the shortest.
3. The green sigil bearer is standing to the left of the black sigil bearer.
4. The 3 foot tall hero carries a club.
5. The fourth hero carries a spear.
6. The hero carrying a sword is next to the hero carrying a club.
7. The lance is not carried by the snake.
8. The wearer of pink armor carries a dagger.
9. The bear is the tallest.
10. The hero beside the wolf carries a sword.
11. 11. The hero in white armor is to the left and next to the hero with black armor.
12. The wolf carries a club.
13. The white sigil is adjacent to the red sigil.
14. The one that wears purple armor carries a spear
15. The wearer of the black armor is 3 feet tall.
16. The owl and the bear guard the outer sides.
17. The bearer of the red sigil is not the tallest
18. The wolf does not wear white armor or a white sigil.
19. The one who wears red armor is 6 feet tall.
20. The 4 foot tall hero is on the outer left and does not like to use swords.
21. The shortest hero carries a club.
22. The snake stands next to the hero with a black sigil.
23. The lion does not wear a black sigil.

K hádance existují právě dvě řešení.