Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Automatizované vyhodnocování dotazníků pro výběr doplňků stravy

Plzeň, 2015 Martin Pausar

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23.června 2015

Martin Pausar

Abstract

The goal of this thesis is building of a pedestal for a knowledge-based system, that will help to automate suggesting relevant products based on lifestyle data entered from a respondent. To bring this idea to life I have chosen Prolog programming language.

At this time the application can handle the input data while reading them from a csv file or being entered through the Prolog language text environment. Entering data by hand is provided by answering questions the system provides. The principle is, the respondent eventually gets information about the suitable food supplements for him or her. And the results are based on combination of the data entered.

Abstrakt

Cílem této práce je vytvoření základů pro systém, který bude pomáhat automatizovat doporučování vhodných produktů na základě zadávání vstupních dat týkajících se životního stylu respondentů. Pro realizaci této myšlenky jsem zvolil programovací jazyk Prolog.

V současné podobě aplikace zvládá základní načítání dat ze souboru csv nebo manuální zadání v textovém prostředí Prologu. Manuální zadávání dat spočívá v odpovídání na otázky, které systém pokládá. Podstatou věci je, že ve výsledku dostane uživatel informace o produktech, které jsou pro něj nejvhodnější na základě kombinace dat, která do systému zadal.

Poděkování

Chtěl bych poděkovat mým nejbližším za trpělivost a podporu při vytváření této práce a v neposlední řadě svému vedoucímu práce Prof. Ing. Václavu Matouškovi, CSc. za inspiraci a cenné rady.

Obsah

1 Úvod

Důvodem pro zvolení tohoto zadání je moje dlouholetá činnost v oblasti poradenství ve výživě. V tomto oboru se konkrétně pohybuji v rámci společnosti, která dováží z Finska doplňky stravy. Tyto doplňky se však zmíněná společnost snaží nejen pouze posunout v distribučním řetězci blíže směrem k zákazníkovi, ale taktéž mu usnadnit výběr.

Jelikož je sortiment velmi široký, čítá řekněme několik set produktů, klade tedy volba náležitého výrobku na zákazníka relativně vysoké požadavky ve smyslu šíře vhledu do oboru výživy. Firma jinými slovy nenechává klienta takříkajíc napospas, aby si sám z široké nabídky vhodný produkt vybral, ale proaktivně hledá mechanismy, které mají sloužit k tomu, aby onen zákazník získal stručnou a jasnou představu o přípravcích, které jsou pro jeho osobu nejvíce žádoucí.

V současné době je výživa celkem exponovanou oblastí a tento zájem radit v oblasti životního stylu a stravování nepřináší pouze pozitiva. Existuje totiž spousta ,,rychlokvašených" poradců, kteří se po absolvování jakéhosi rychlokursu pouštějí hned druhý den do léčení nejtěžších chorob.

Na druhou stranu i právě vzhledem k velkému zájmu veřejnosti o tento obor spatřují světlo světa různé způsoby volby toho nejvhodnějšího preparátu nebo jejich kombinace pro konkrétního spotřebitele. Jedním z mnoha těchto způsobů je získání náležité palety vstupních informací pro jejich pozdější vyhodnocení. A to například formou výstižně zvoleného dotazníku.

Zadání práce vychází právě z takovéhoto dotazníku. Tento je tedy, stručně řečeno, koncipován pro orientační doporučení vhodnosti užívání konkrétních doplňků stravy takzvaně ,,na míru" pro konkrétního jednotlivce, který dotazník vyplňuje. Jedná se zde opravdu o doporučení orientační, protože při vyhodnocování rezignujeme na jakákoli upřesnění dat, která respondent do dotazníku vyplní. Tím je myšleno vypuštění jakýchkoli zpřesňujících měření nebo vyšetření.

Výstup z takto získaných a následně vyhodnocených údajů je však přesto pro užitek koncovému uživateli podstatně přínosnější, než orientace podle televizní reklamy, doporučení od souseda nebo využití dalších podobných více či méně obecných až zavádějících zdrojů informací.

Zadávání dat probíhá v současné době přes webové rozhraní s tím, že zadané údaje se ukládají do tabulky, ze které se pak hromadně odesílají ke zpracování. Alternativně lze dotazník vyplnit i v tištěné formě. V dalším textu je popisován dotazník v aktuální podobě. Pro potřeby aplikace bude rozhraní zajišťující zadávání dat upraveno.

Odeslaná data se v současné době vyhodnocují způsobem postaveným na základě odhadu životního stylu respondenta podle obsahu a struktury vyplněných údajů. Vyhodnocení a výstup v podobě doporučení vhodného doplnění stravy jednotlivými doplňky se aktuálně provádí na anonymní bázi některým z lékařů, odborníků na výživu, kteří jsou podrobně seznámeni s deklarovanými účinky zmíněných doplňků

stravy.

V rámci současného stavu vyhodnocování mohou být výstupní data obohacena o doporučení z hlediska změn v jídelníčku respondentů respektive naznačení vhodné formy pohybové aktivity. Tyto informace ve výstupu zamýšlené aplikace zahrnuty nebudou.

Mým cílem je postupné vytváření aplikace, která bude v rámci možností plnit funkci těchto odborníků a poskytovat výsledky respondentům v kratším časovém intervalu a činit je tedy dostupnějšími a nezávislými na časových možnostech jednotlivých odborníků.

Rozhodl jsem se vyhodnocování dotazníku implementovat jako znalostní systém. Budování takovýchto systémů je dlouhodobá záležitost, proto je již teď jasné, že ani po dokončení a odevzdání této práce nebude možno považovat systém za finální. Nicméně předpokládám, že bude zřejmé, jakým způsobem jej chci dále rozvíjet, jak je myšlena jeho obecná funkčnost a že celková koncepce dává smysl.

Dále předpokládám, že i pouhé jádro systému ve formě základní báze znalostí propojené s možnostmi jazyka Prolog bude použitelné pro demonstraci možností rozšíření této báze a případné přenositelnosti navržených pravidel systému do systému podobného. Tím myslím možnost použití i pro jiné firmy v oboru.

2 Specifikace zadání

2.1 Obecný popis vstupních dat a jejich zadávání

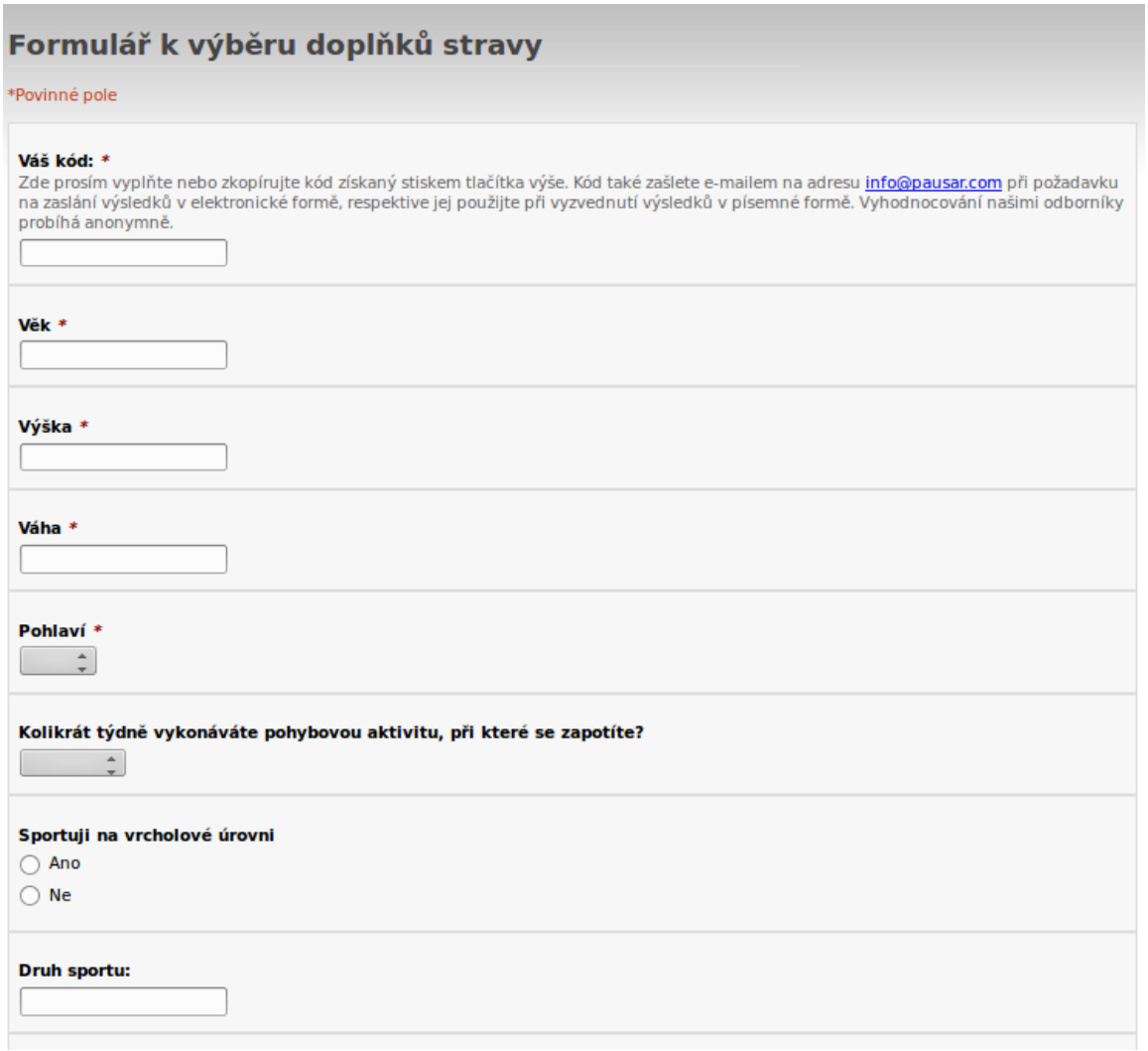
Dotazník je složen z bloků otázek, které by měly při zachování relativní stručnosti tohoto dotazníku vystihnout životní styl respondenta s akcentem na jeho stravovací návyky. Tím je myšlena jak pravidelnost stravování tak i hrubá skladba jídelníčku. Při vyplňování se proto nezaměřujeme na konkrétní rozbor toho, jak se respondent stravuje - v granularitě jednotlivých jídel, respektive jejich výživové hodnoty - nýbrž pouze na skupiny potravin.

To v praxi znamená požadavek na získání informací o tom, jak často se na stolech strávníků, jejichž údaje chceme vyhodnocovat, objevují např. ryby, ovoce a zelenina, červené maso atd. Podobně je tomu u pitného režimu, kde se vychází pouze z převládajícího nápoje, jinými slovy uvede se do odpovědi ten nápoj, kterého respondent vypije dle svého názoru nejvíce.

2.2 Antropometrické[1] údaje a pohybová aktivita

Základními údaji, ze kterých můžeme vycházet a které mají již samy o sobě nemalou signifikantní hodnotu, jsou výška, váha a věk respondenta. Na základě informací o výšce a váze můžeme snadno vypočítat tzv. BMI index, který má relativně slušnou vypovídací schopnost ohledně hodnocení nutričního stavu respondenta,

[1] týkající se měření lidského těla (jeho rozměrů atd.)



Obrázek 1: Část formuláře pro Antropometrické údaje a pohybovou aktivitu

jehož vstupní data chceme vyhodnocovat.

U většiny lidských jedinců lze při zevrubném pohledu určit, zda se jedná o muže nebo ženu a s minimem životních zkušeností také zjistit, že existují určité rozdíly ve stavbě těla každého z nich (i když jsem se setkal s názory několika militantně extremistických členů(ek) různých hnutí, že to tak vlastně není) a navíc v lidském těle existují procesy, které probíhají jinak u mužů a jinak u žen. Proto je vhodné do takovéhoto dotazníku dle mého názoru zařadit i otázku na pohlaví respondenta.

V tomto bloku jsou ještě požadovány informace o druhu a intenzitě pohybové aktivity vykonávané takzvaně do zapocení. Pro jednoduchost vstupu opomíjíme jakákoli měření v průběhu těchto pohybových aktivit - např. měření tepové frekvence, na základě které dokážeme podstatně lépe doladit vhodnou délku a intenzitu cvičení, aby bylo pro organismus co nejefektivnější. První část dotazníku věnovanou zjišťování výše zmíněným skutečnostem můžete vidět na obrázku č.1.

2.3 Užívání léků a povolání

Pokud respondent užívá léky, uvádí to taktéž při vyplňování dotazníku. Pokud je odpověď kladná, znamená to, že respondent s největší pravděpodobností řeší určité zdravotní problémy. Což nám ztěžuje situaci, jelikož další vyhodnocování by bylo vhodné konzultovat s lékařem.

Pro převedení vyhodnocení vstupních dat do elektronické podoby je třeba některé otázky co nejvíce zobecnit, aby odpověď na ně byla alespoň určitým způsobem vypovídající. A to se týká mimo jiné právě například užívání léků. Můžeme totiž v tomto případě uvažovat o doporučovaném doplňku stravy v souvislosti s medikací respondenta ze dvou základních směrů. Jsou tím samozřejmě myšleny směry vedoucí k nějakému užitku pro respondenta.

Za prvé lze určité doplňky stravy používat jako podporu léčby, pro kterou je indikován konkrétní lék, což nutně znamená minimálně získání dalších doplňujících informací. Rozhodl jsem se tento směr vynechat. Nejen pro složitost získávání těchto doplňujících informací, ale taktéž pro nutnost konzultace s ošetřujícím lékařem, který podrobně zná zdravotní stav respondenta.

Druhý směr je povšechná ochrana před škodlivými účinky léků jako takových. A z toho se při dobré vůli relativně obecný závěr vyvodit dá.

Jelikož téměř všechny léky při pravidelném užívání zatěžují játra, zařazuje se do výstupu některý z preparátů, jemuž jsou přisuzovány účinky pro ochranu tohoto důležitého orgánu. Samozřejmě v určitých případech s poznámkou o nutnosti dodržení časového rozmezí minimálně dvě hodiny mezi užitím léku a doplňku stravy.

Při klasickém vyhodnocování dotazníku odborníkem uvádí respondent ještě druh léku a dávkování. Vzhledem k počtu léků a diagnóz jako takových je velmi obtížné tuto nadstavbovou informaci zahrnout do automatizovaného vyhodnocení a proto jsem se taktéž rozhodl ji prozatím vypustit.

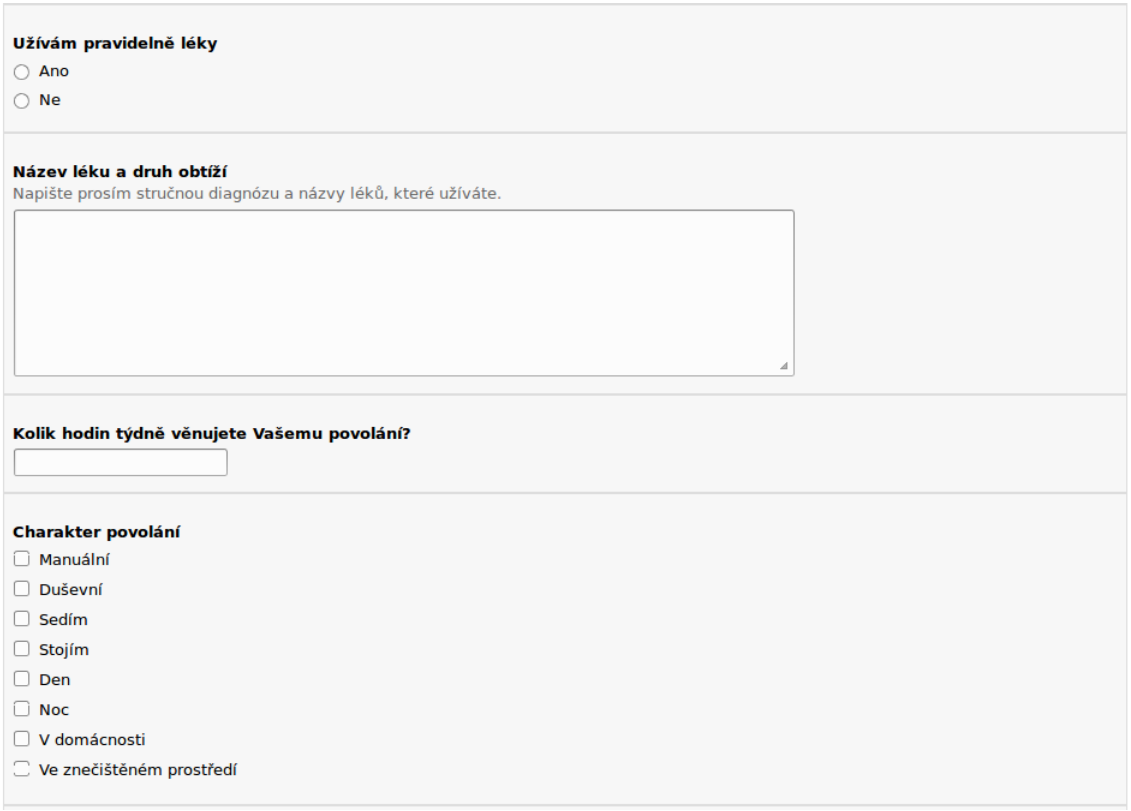
Důležitý je pochopitelně také údaj o zaměstnání respektive povolání respondenta. Zjišťuje se celkový počet odpracovaných hodin týdně a také charakter povolání. Tím je myšleno, zda respondent pracuje fyzicky nebo duševně nebo zda při práci převážně sedí nebo stojí.

Ohled je brán taktéž na to, zda práce probíhá ve dne či v noci a zda respondent pracuje ve znečištěném prostředí (např. zvýšené množství prachu nebo nadstandardní přítomnost chemikálií ve vzduchu) nebo je vystaven nadměrnému hluku, teplotě, záření a podobně (viz obrázek 2).

2.4 Kouření, alkohol a spánek

Existuje však mnoho dalších faktorů, které mohou ovlivňovat zdravotní stav respondenta a některým z nich je věnována další část dotazníku.

Patří mezi ně například množství konzumovaného alkoholu, který má od určitého kvanta negativní vliv opět na játra a samozřejmě na mozek a pak také kouření, což je v kontextu ovlivnění zdravotního stavu jedince kapitola sama pro sebe.



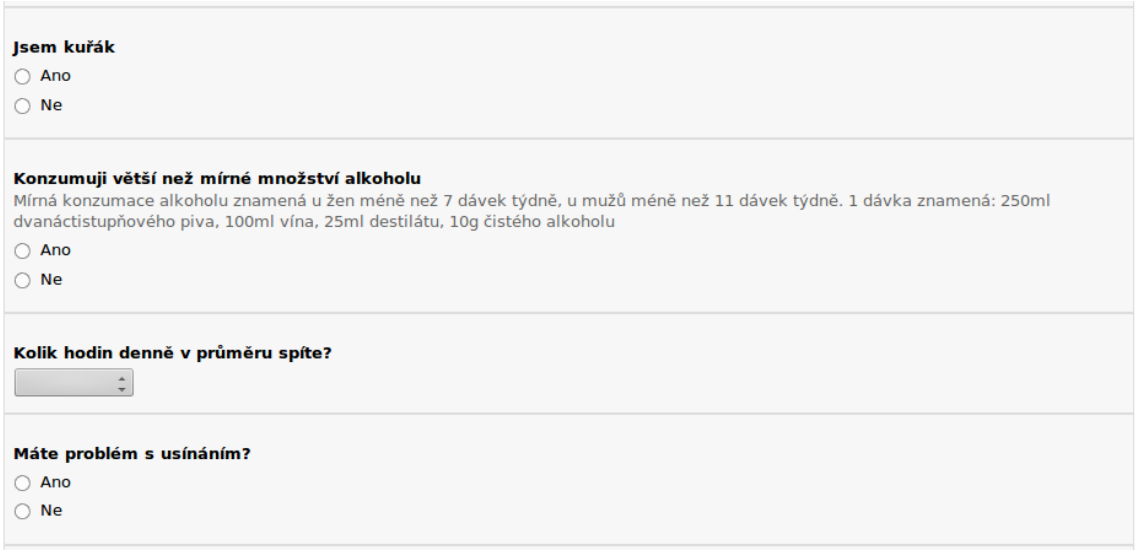
Obrázek 2: Část formuláře pro zadání dat ohledně užívání léků a povolání

Výzkumem zjištěných negativních vlivů kouření je prokazatelně mnoho. Zásadní je samozřejmě vliv na zvýšený výskyt srdečně cévních onemocnění. Doplnění vitamínu C může částečně některé škodlivé účinky kouření eliminovat.

Tím však není řečeno, že pokud člověk kouří a zároveň užívá vitamin C, nemusí se obávat rizik, která s sebou kouření přináší. Kuřák užívající dostatečné množství vitaminu C má pouze naději, že jeho zdraví bude devalvovat při dobré konstelaci ostatních faktorů ovlivňující jeho zdravotní stav pomaleji, než kdyby tento vitamín nebyl v jeho organismu zastoupen v dostatečném množství.

Důležitým kouskem skládanky, ze které v našem případě má vyústit orientační seznam preparátů - doplňků stravy, které by mohly dotyčné osobě vyplňující dotazník být prospěšné pro přiblížení se požadovanému stavu tj. dlouhodobě trvajícímu správnému fungování organismu, patří bezesporu také spánek.

V dotazníku je otázka předestřena jako počet hodin, které v průměru za týden respondent naspí (viz obrázek 3). Kvalita spánku má samozřejmě velký vliv na množství energie, které může respondent denně spotřebovat aniž by pociťoval nadměrnou únavu. Je více než jasné, že spánek žádný doplněk stravy nenahradí. V kontextu s doplňující otázkou ohledně problémů s usínáním je však možné doporučit produkt, který může pomoci k upravení tohoto stavu směrem k normálu nebo požadovanému stavu.



Obrázek 3: Část formuláře pro data související s kouřením, alkoholem a spánkem

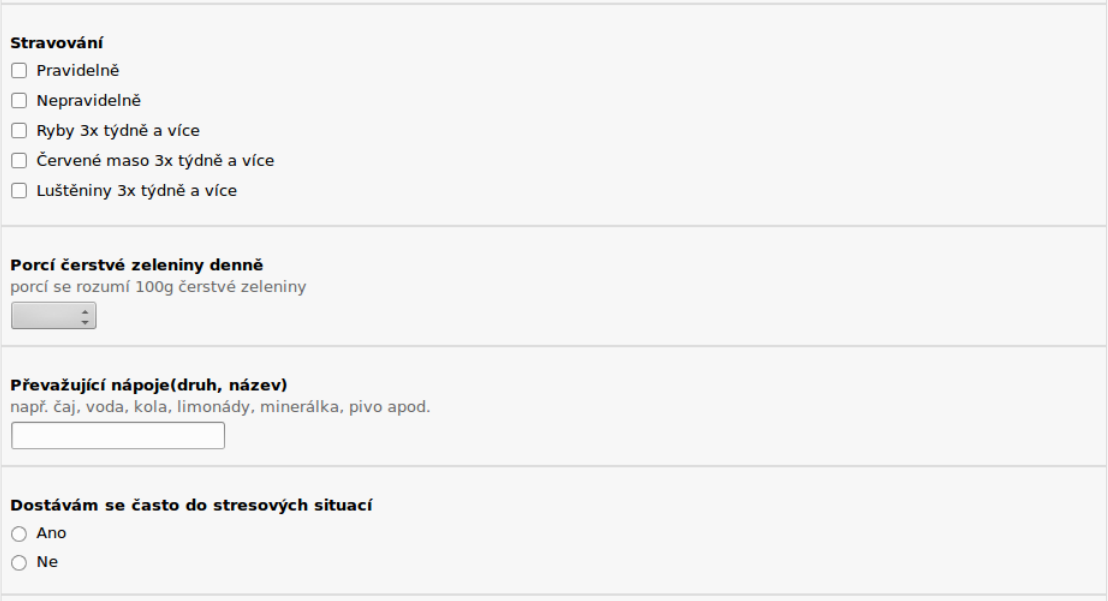
2.5 Strava, nápoje a stres

Pro získání orientačního náhledu na životní styl respondenta potřebujeme vědět alespoň hrubé rysy jeho jídelníčku. Tomuto tématu se věnujeme počínaje blokem „Stravování". Pravidelnost stravy je velmi důležitá. I proto je v dotazníku zahrnuta. Tím je myšlena pravidelnost v několika souvislostech. Jedna z nich je vztažena k denní době, ve kterou se stravujeme nebo dokonce ke konkrétní hodině. Někdo například tvrdí, že když se nenaobědvá přesně v poledne, má pak celý den žaludek jako na vodě. Není to však jen pouhé tvrzení laiků. K tomu, že se jedná o důležitou věc, se přiklánějí i někteří odborníci. Jiní naopak soudí, že důležité jsou intervaly mezi jednotlivými jídly a vykládají termín pravidelnost právě tímto způsobem. Tělo podle nich vyžaduje přísun energie každé tři až čtyři hodiny. Pokud se tento rozestup zvětší, organismus si vytváří zbytečné zásoby a lidé pak tloustnou, i když celkově toho sní v podstatě stejně. Setkal jsem se i s názorem, že je důležitější nejdříve začít jíst pravidelně a teprve pak se starat o to, zda to, co jíme splňuje direktivy zdravé výživy.

V rámci této pravidelnosti samozřejmě není možné v plné míře nahradit některé z hlavních jídel doplňkem stravy, ale je možné nahradit nebo doplnit svačinu např. tabletami z některých řas a nebo překlenout v ojedinělých případech nedodrženou pauzu mezi jídly, která by při dodržení pravidelného jídelníčku neměla přesáhnout 3 hodiny.[2]

V neposlední řadě se taktéž zjišťuje zatíženost respondenta stresem. V základu je uvedena otázka, zda se dotyčný dostává, dle svého subjektivního dojmu, do stresových situací často nebo méně často. Volitelně je pak možné absolvovat doplňkový test na zvládání působení stresu[3]. Jedná se o tzv. Bortnerovu krátkou škálu. Tato

[2]http://www.berkshirehealthcare.nhs.uk/_store/documents/regular-eating.pdf
[3]http://www.szu.cz/tema/podpora-zdravi/hodnoceni-psychosocialnich-faktoru



Obrázek 4: Část formuláře pro vyplnění údajů o stravování, nápojích a stresu

metoda spočívá ve vyplnění sebe-posuzujícího dotazníku, který je pak vyhodnocován Státním zdravotním ústavem. Vyhodnocování probíhá automaticky pomocí počítačového programu. Existují funkční odlehčené open source varianty k tomuto vyhodnocení stresové zátěže, které by se za určitých podmínek daly implementovat do konečné fáze mého programu (viz obrázek 4).

Dále je pak v dotazníku textové pole pro uvedení stručné anamnézy respondenta a tedy k popsání chorob svých nebo rodičů, z čehož pak může vyplynout příznivé působení určitých účinných látek obsažených v preparátech ze sortimentu, který má být výstupní informací vyplněného dotazníku. V tomto případě by však u implementace automatizované formy vyhodnocování mohl nastat problém v nejednoznačnosti interpretace anamnézy jednotlivými respondenty. Tato situace se dá v rámci znalostních systémů částečně řešit, nicméně tento bod jsem se taktéž prozatím rozhodl vypustit.

Při konzultaci s experty, kteří nám dotazníky vyhodnocují jsem se setkal s tím, že jejich oblíbený vstup je vyjádření vlastního názoru respondentů. Mezi otázkami je proto ponechán respondentům prostor pro posouzení svého zdraví. Otázka je pro urychlení vyplňování formulovaná formou výběru z několika zásadních oblastí, ve kterých je dle názoru vyplňující osoby potřebné podpořit její zdravotní stav. Mezi jinými volbami zde najdeme například pokyny k posouzení vlastní vitality, kvality spánku, míry spokojenosti se stavem vlasů, nehtů a pleti. Respondenta můžeme nechat sama sebe ohodnotit vlastní výkon v souvislosti s pamětí a soustředěností. Takových možností je samozřejmě celá řada. Je však třeba vybrat rozumné množství, abychom přehnanou podrobností respondenta neodradili od dalšího pokračování ve vyplňování dotazníku. Proto jsem si opět nechal poradit od odborníků a zvolil ty možnosti, které se objevovaly při klasických konzultacích, kdy zájemci o radu

pokládali otázky lékařům buď osobně, e-mailem nebo telefonicky.

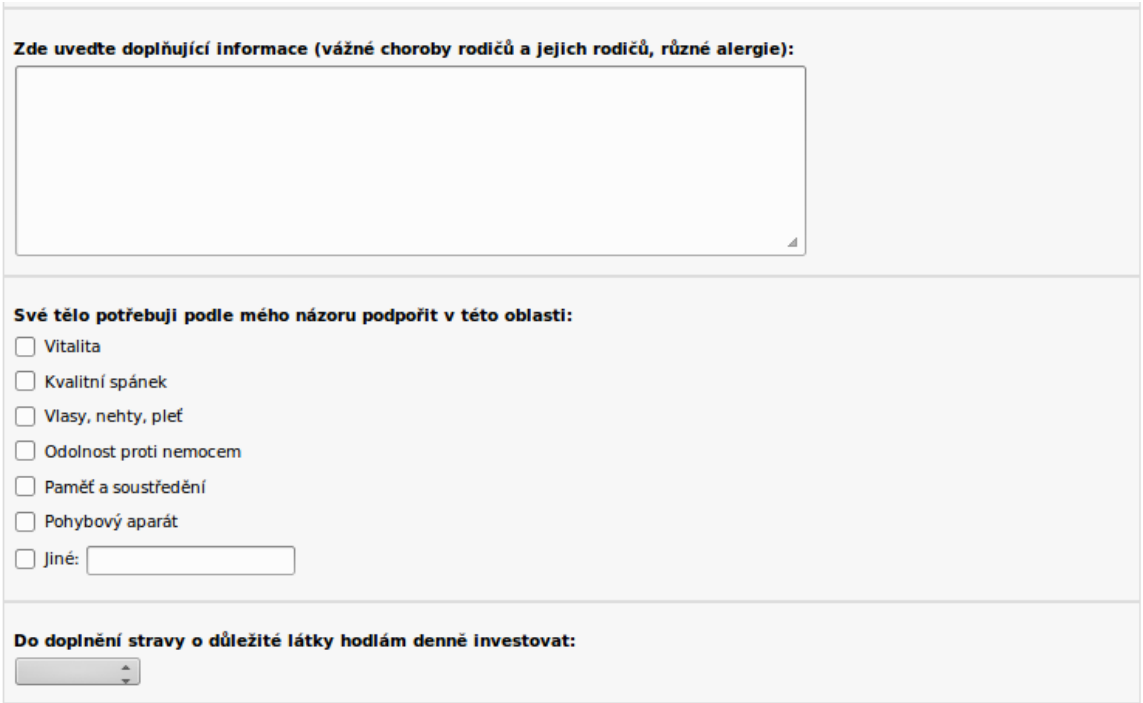
V rámci klasického vyhodnocování je jako jedna z možností stanovení vlastní oblasti podpory zdraví, kdy respondent sám vepíše v jakém směru by dle svého názoru potřeboval zdravotní stav podpořit. Tato možnost však bude narážet na podobné problémy s nejednoznačností jako je uvedeno výše u popisu anamnézy.

Poslední dvě informace, které chceme od respondenta vědět jsou datum, kdy dotazník vyplňuje, ze kterého pak vychází sezónní doporučení doplnění stravy, a pak také suma, kterou je ochoten do doplňků stravy investovat.

Na základě data tak může být třeba v létě doporučen beta-karoten, který může pomoci zabránit tomu, abychom na slunci zčervenali (nepomůže však v případě, že přijedeme na dovolenou do tropického pásma a hned od prvního dne budeme celý den ležet na slunci). Na podzim lze doporučit například některý z preparátů podporujících imunitu.

Využití data by se dalo dál rozšířit například tím, že k vlivu ročního období na výstup dotazníku, by bylo samozřejmě přesnější provázat výstup s předpovědí počasí a doplnit otázkou na místo pobytu např v příštích 14 dnech, což by jednak bylo technicky náročnější, ale nikoli neřešitelné. Problém vidím spíše v zásahu do soukromí respondenta.

Podle vyplněné sumy, kterou je respondent připraven utratit, je velmi ovlivněn celkový výstup dotazníků. Pokud by tam bylo například vyplněno 1 Kč za den, výstup by nebyl v podstatě žádný. Proto jsou možnosti předvyplněné od minimální sumy až do průměrné a vyšší. (viz obrázek 5).



Obrázek 5: Část formuláře určená pro anamnézu, vlastní pohled a závěrečné informace

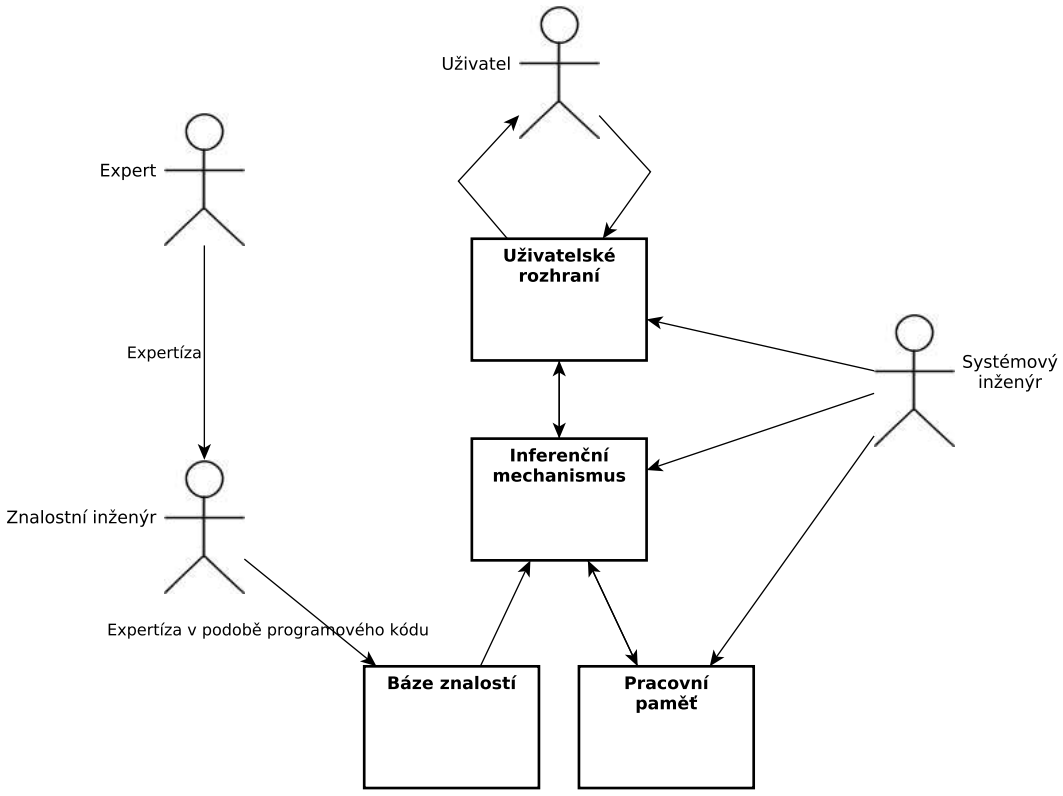
3 Postup převodu dotazníku do automatizované podoby

3.1 Znalostní systémy

3.1.1 Obecný úvod

Zde je na místě zmínit se alespoň stručně o tom, jak takový znalostní systém vypadá a jaká forma implementace by byla pro tento případ nejvhodnější.

Znalostní systémy jsou počítačové programy, používané pro řešení takových úloh, které jsou všeobecně obtížné a jejichž uspokojivé řešení může provést pouze specialista (expert) v daném oboru. Dříve se používal termín ,,Expertní systém“. V současnosti však zpřístupnění technických prostředků čím dál širší veřejnosti a snižování cen komponent, jejichž výkon postačuje na provozování takovýchto systémů postupně sjednotilo jejich označení jako znalostní. To znamená, že lze použít i znalosti obecnějšího rázu. Jinými slovy nemusíme se ptát na data, která bude náš systém používat, jen experta, ale můžeme například využít znalostí a zkušeností dalších lidí, pohybujících se více či méně v oboru, který chceme naším systémem obsáhnout.



Obrázek 6: Součásti znalostního systému

Znalostní systémy se používají kupříkladu pro diagnostické postupy, které se dají přenést do automatizované podoby. Jejich výstup pak může sloužit jako vodítko dalšího postupu pro člověka, který na jeho základě provede nějakou následnou

činnost nebo také jako vstup pro další automat, který dle obdržených výstupních hodnot, začne vykonávat předem určenou posloupnost kroků v rámci svého pracovního nasazení. Znalostní systém ale může také sloužit například k tomu, aby byl tréninkovým nebo někdy i soutěžním partnerem ve hře v šachy.

Na obrázku 6 jsou vidět hlavní součásti znalostního systému a cesta informací, které jednak musíme do systému dostat a pak je přemodelovat v odpověď pro uživatele, který se systémem pracuje a používá výstupní data pro svoji další činnost.

V první fázi je nutno provést konzultace s expertem (i když budu tuto osobu označovat jako experta, může jít, jak již bylo výše řečeno o kohokoliv, jehož dovednosti a zkušenosti jsou pro náš budoucí systém natolik cenné a významné), a získat od něho potřebné znalosti a postupy, kterými tyto znalosti převádí ve své konečně rozhodnutí v rámci řešeného problému. Znalostní inženýr musí na základě těchto konzultací pokud možno převést znalosti a postup činnosti experta do deklarativní formy, použitelné pro vytvoření báze znalostí. Popíše tedy vhodnou formou to, co expert ví a jak se dopracovává k výsledku.

Systémový inženýr pak na základě tohoto popisu zpracuje konkrétní návrh znalostní databáze a jejích pravidel. Dále vytvoří uživatelské rozhraní, díky kterému pak se systémem komunikuje koncový uživatel a v neposlední řadě také provede implementaci vybraného inferenčního mechanismu.

Pak je možné, aby koncový uživatel zadával systému dotazy. Inferenční mechanismus pak může poskytovat kýžené odpovědi na základě pravidel a faktů, která byla při vytvoření systému zadána jeho tvůrci, ale také dál plnit bázi znalostí informacemi, které vzniknou při zpracovávání toho, co již je známo.

Pokud se vše podaří, měl by pak počítač pomocí svého inferenčního mechanismu uvažovat (odvozovat) stejným způsobem jako expert.

Složitost vyhodnocovaného dotazníku není možná úplně na první pohled patrná, nicméně k jeho vyhodnocení je v současné době využíváno lékařů, kteří se specializují na oblast výživy. Tito si na vyhodnocení dotazníku musejí vyhradit určité penzum času a jejich dostupnost není vždy dostatečná do té míry, do jaké je právě v daném okamžiku potřeba, což způsobuje prostoje a jak víme, zákazníci neradi na něco čekají.

Ve světle těchto okolností jsem přesvědčen, že tento projekt v zásadě požadavky na znalostní systém splňuje.

3.1.2 Pravidlové systémy

Pravidlové znalostní systémy se vyznačují tím, že znalosti v nich obsažené jsou reprezentovány jako databáze pravidel. Tato databáze spolu s fakty obsaženými v pracovní paměti dodává inferenčnímu mechanismu potřebné informace, na jejichž základě je pak pro takto koncipovaný znalostní systém možné odpovědět na zadaný dotaz.

Odpovědět samozřejmě v tomto kontextu neznamená poskytnout při každé příležitosti takový výstup, jaký uživatel od systému očekává. To, že se po zadání dotazu díky komunikaci přes uživatelské rozhraní dozvíme, že systém na námi zadaný dotaz

nedokáže odpovědět, je však v podstatě také správná odpověď. Neznamená to tedy, že systém nefunguje, jen je třeba případně doplnit jeho bázi znalostí.

U pravidlových systémů je princip činnosti založen na platnosti či neplatnosti určitých předpokladů a inferenčním mechanismem generovaných závěrů z těchto předpokladů plynoucích. Aplikují se tedy pravidla ve tvaru *jestliže* $\rightarrow$ *pak*. Více se pravidlům budu věnovat v sekci popisující implementaci. Podrobnější informace lze taktéž nalézt v [6].

Pravidlem vyjádřený výraz tak představuje pozorování posloupnosti jevů, které za předpokladu, že nastanou nebo nastaly, vedou na vyslovení hypotézy, která se v závěru pravidla buď potvrdí nebo ne.

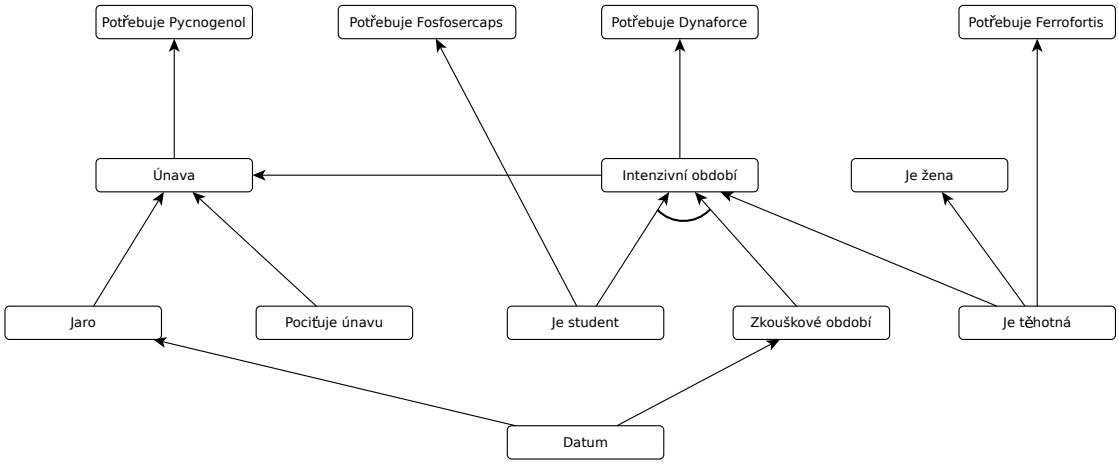
Odvozování, usuzování, nebo-li inference, je v rámci pravidlových znalostních systémů založena na pravidle modus ponens. Vyjádřit se dá tímto vzorcem:

$$\frac{E, E \rightarrow H}{H} \tag{1}$$

Slovně vyjádřeno to znamená, že pokud platí předpoklad E a pravidlo $E \rightarrow H$, pak platí i závěr H. Modus ponens vychází ze slova ponere, což znamená tvrdit. Je to typ usuzování, který nazýváme usuzováním přímým. Naopak usuzování nepřímé je vyjádřeno pravidlem modus tollens, jehož význam má původ ve slově tollere, což znamená popřít:

$$\frac{\neg H, E \rightarrow H}{\neg E} \tag{2}$$

Význam (2) můžeme popsat tak, že pokud se nepotvrdila hypotéza H a současně platí pravidlo $E \rightarrow H$, pak byl předpoklad E nesprávný.



Obrázek 7: Inferenční síť

Na základě rámcového prostudování možností znalostních systémů a vzhledem k povaze řešeného problému jsem proto zvolil právě použití pravidlového systému implementovaného jako inferenční síť (viz obrázek 7).

Je zde k náhledu výřez konkrétní inferenční sítě použité v mojí aplikaci. Jednotlivé uzly představují buď fakta nebo hlavičky pravidel. Šipky pak znázorňují, jakým způsobem a jakým směrem se daná pravidla vyhodnocují.

Pokud jsou šipky spojené obloučkem, vyjadřuje to logický součin nebo jinak formulováno logickou spojku AND. Pokud obloučkem spojené nejsou, znamená to logický součet potažmo spojku OR.

Pokud bychom se chtěli vyjádřit co nejstručněji, vidíme z této části sítě, že, v nadsázce řečeno, ideálním potenciálním zákazníkem by byla těhotná studentka v rámci zkouškového období.

Mimo jiné je zde také vidět, původně neplánovaný ale veskrze pozitivní vedlejší účinek aplikace, a to konkrétně, že by tento systém mohl být využit nejen pro získání informací, jaký preparát je pro toho kterého jedince v aktuální situaci nejvhodnější, ale i pro účely marketingu, kdy nám za určitých okolností může ukázat, jaké zákazníky vyhledávat.

Tento jednoduchý znalostní systém, jehož složitost a potažmo i využitelnost hodlám postupem času navyšovat, bude patřit mezi tzv. problémově orientované. Báze znalostí bude již po vytvoření tohoto systému naplněna daty z příslušného oboru. Nebude se tedy jednat o systém prázdný neboli shell, který naopak ve výchozí pozici obsahuje pouze kostru systému a je do něho třeba data zadat, aby byl pro pozdějšího uživatele použitelný a bylo možno získat požadované odpovědi z požadované domény.

Popsané systémy se dají realizovat několika rozličnými způsoby v některém z deklarativních programovacích jazyků, ze kterých se nabízí třeba Prolog. V úvahu by mohlo připadat také použití jazyka Lisp nebo využití některých již hotových prázdných (shell) znalostních systémů. Já jsem se rozhodl pro Prolog.

3.1.3 Postup vyhodnocování experta

Znalostní systém by, dle své definice, měl simulovat jednání nebo postup experta při řešení zadané úlohy. Proto jsem na toto téma konzultoval s lékařkou v oboru hygiena výživy MUDr. Vladimírou Kohoutovou, která dotazníky prozatím ,,ručně“ vyhodnocuje a má v tomto ohledu již mnohaletou praxi.

Začíná vždy s vyhodnocením případné nadváhy respondenta, od které se pak mohou odvíjet různé další indikace, které zatím nebudou pokryty automatizovanou verzí. Tento postup pak pokračuje dál v souvislosti s pohlavím respondenta, které může být taktéž vztaženo k věku, což může opět dávat odpovědi ještě než jsou položeny konkrétní otázky.

Dále potom přichází na řadu konkrétní zdravotní obtíže, ze kterých se již může konkrétně doporučovat jeden nebo několik produktů pro podporu léčby.

Tento postup mi však připadal pro konverzi do automatizované podoby příliš ,,lidský“. Zvolil jsem proto opačný směr. Je to dle mého názoru směr vhodný i s ohledem na případnou budoucí komerční využitelnost aplikace.

Ve výsledku totiž potřebujeme víceméně vědět, jaký produkt nebo produkty jsou pro respondenta vhodné. Z toho jsem vycházel a jako základní prvek použil jednotlivé preparáty. Těch je sice v sortimentu hodně, ale jejich množství je podstatně omezenější než počet různých symptomů, kdy ještě navíc každý sám o sobě může znamenat několik odlišných diagnóz. O jejich kombinacích ani nemluvě.

Proto se tedy stručně řečeno v případě automatizovaného vyhodnocování probíráme buď sortimentem celým nebo v našem případě pouze jeho výsečí, což je pro ukázku funkčnosti a použitelnosti aplikace dostačující.

Vezmeme tedy například preparát s vitamínem C a na základě dat vyplněných do dotazníku určíme, jestli je pro daného respondenta v aktuální době vhodný či nikoli. A tak pokračujeme dál, dokud nám zbývají nějaké produkty, které jsme v tomto směru ještě neprozkoumali.

Tento postup by pro člověka byl podstatně časově náročnější, kdežto počítači je to při dnešních výpočetních výkonech prakticky jedno.

4 Prolog

4.1 Prolog k Prologu

Prolog je označován jako jazyk deklarativní, což znamená, že použitím jeho konstrukcí formulujeme problém pomocí predikátové logiky prvního řádu.

Mé první setkání s tímto jazykem budilo spíše rozpaky. Možná proto, že do doby, než jsem se začal o Prolog poprvé zajímat, jsem přišel do styku pouze s jazyky procedurálními. Konkrétně to začínalo Javou a pak pokračovalo přes C a C#. Zmiňuji se zde o tom z toho důvodu, že mám v úmyslu na dalších stránkách přistupovat k Prologu spíše jako k nástroji, který se člověku mění pod rukama v závislosti na tom, jaký pro něj vymyslí úkol, než striktně jako k programovacímu jazyku.

V žádném případě se v této chvíli necítím být expertem na Javu, C nebo C# natož na Prolog. To předesílám proto, že rozhodně nechci posuzovat nebo porovnávat míru použitelnosti zmíněných jazyků. Nicméně Prolog ve mně vyvolává zatím největší sympatie, které do té doby držel vcelku bezpečně v rukou jazyk C. Tento zajímavý počin, který balancuje na hraně mezi programováním a správným popisem řešeného problému mne nutí se jím zabývat stále hlouběji. Je to možná způsobeno tím, že jsem již od malička inklinoval k sci-fi filmům a u těch, které se věnovaly tématu umělé inteligence jsem doslova propadal nadšení. Kromě filmů však čerpám i ze spousty zajímavých publikací na toto téma jako třeba [2] nebo [7]. Zatím mi bohužel čas nedovolil věnovat se získávání dalších informací v této oblasti v rozsahu, který bych považoval za adekvátní mému zájmu, ale to chci v budoucnu změnit. Jednou z prvních souvislejších příležitostí je právě tato práce, kdy jsem mohl skloubit svůj obor činnosti s tímto mým oblíbeným tématem. Při volbě zadání jsem byl pevně přesvědčen, že to může být začátek vzniku zajímavé a užitečné aplikace, kterou budu stále vylepšovat.

Prolog má sám o sobě tu výhodu, že již obsahuje mechanismus pro práci s inferenční sítí, kterou vytvoříme popsáním řešené situace pomocí faktů a pravidel. V případě použití tohoto jazyka se již nemusíme zabývat vymýšlením a vytvářením procedur a funkcí, které počítači vysvětlují, jak má dojít ke svému cíli.

Deklarováním faktů a pravidel, zmíněných v podsekci 3.1.2 o pravidlových systémech obecně, vytvoříme mikrosvět postavený na cíli, ke kterému chceme dojít a na cestě, kterou se k němu musíme vydat. Přičemž se ani tak nezabýváme tím, že bychom si tuto cestu nově prosekávali nějakou džunglí, kde ještě nikdy nikdo nebyl, ale spíš hledáme již existující kousky cest, které se snažíme napasovat na sebe tak, abychom se dopracovali tam, kam se nakonec dopracovat chceme a abychom od takto sestaveného programu dostali takovou odpověď, která nám pomůže v našem dalším rozhodování nebo v ideálním případě udělá rozhodnutí za nás.

U Prologu lze navíc zvolenými cestami procházet tam i zpět. Jinými slovy řečeno při brouzdání se inferenční sítí lze použít dopředné i zpětné řetězení, což také vyhovuje implementaci zadání, které je v rámci této práce řešeno.

4.2 Báze znalostí a její průběžné doplňování

Má představa o bázi znalostí a souvislostech s ní spojených se ucelovala v začátku díky informacím z [5] a v dalších fázích práce také např. z kapitoly *Reprezentace znalostí* obsažené v [2].

Program v Prologu je vlastně báze znalostí zapsaná do textového souboru s příponou dle použité implementace. Báze znalostí popisuje v reálném čase současný stav řešené úlohy. Její část může být trvalá, to znamená, že představuje trvale platné skutečnosti a další část může být výsledkem provedených pravidel, díky kterým se vytváří nová data v rámci této báze.

Konkrétně báze znalostí v Prologu je množinou faktů a pravidel. Fakta jsou statické informace, které mají vypovídací hodnotu sama o sobě, jinými slovy - jsou vždy platná. Pravidla jsou vyjádřena formou podmínek. Jestliže je splněna podmínka, pravidlo platí.

Vznesením dotazu prochází inferenční mechanismus postupně všechna pravidla, která jsou v bázi zanesena právě pro popis tohoto dotazu. Při splnění všech předpokladů je buď pravidlo považováno za platné a uživatel je o tom informován nebo může být báze znalostí doplněna o další fakt, vycházející z platnosti úspěšně nebo i neúspěšně unifikovaného pravidla.

Jak jsem již naznačil v odstavci popisujícím přístup experta k řešení problému, jehož automatizovanou podobu se snažím touto prací nastartovat, v takto koncipovaném znalostním systému vidím tři základní cesty, které vedou k cíli ve formě levnější a rychlejší alternativy k využívání služeb zmíněného experta.

První spočívá v tom, že se do báze znalostí na začátku zadají všechna známá data od uživatele. Tím se doplní báze znalostí o další fakta a pravidla. Na základě těchto doplněných informací a informací již dříve v bázi obsažených pak může dojít

k získání požadovaného výsledku nebo návrhu dalšího postupu.

Druhý přístup vychází z priorit, na základě kterých jsou informace v databázi seřazeny. Pravidla, která nás mají vést k rozhodnutí, pro které nám náš znalostní systém poskytne podporu, mohou být v bázi seřazena buď v libovolném pořadí nebo, pokud se objeví určité omezující podmínky, které si vynutí stanovení priorit pro jednotlivá pravidla, v pořadí reflektujícím právě existenci těchto omezujících podmínek.

Kombinací různých kritérií a priorit je samozřejmě nepřeberné množství, takže tato možnost není úplně triviální z hlediska obecného, nicméně pokud budeme mluvit v dalších částech mé práce již konkrétně, dá se jejich okruh podstatně zúžit.

Třetí způsob spočívá v použití aplikace jako black boxu, kdy se na vstup přivedou data (např. ve formě souboru, obsahujícího hodnoty oddělené čárkami), nechají se aplikací zpracovat a výstup je pak tvořen dalším souborem (například textovým), který pak může vyhodnocovatel dále zpracovávat.

4.2.1 Fakta

Všechna klíčová fakta z báze znalostí budou popsána v programové dokumentaci. Zde bych chtěl jen stručně nastínit o co se jedná. Základní informace se objevila také již v předchozím odstavci.

Fakta jsou ve své podstatě statické informace, které se v průběhu práce s bází znalostí nikterak nemění. Jedná se většinou o parametry jako například jak jsou některé věci velké, staré, výkonné nebo o jejich stav, který je natolik trvalý, že se nebude měnit během očekávané doby používání té které aplikace.

4.2.2 Pravidla

Pravidla nás posunují k požadovanému výsledku. Jak již bylo řečeno v minulém odstavci a také v sekci obecně popisující pravidlové systémy, pravidla mohou mít a v Prologu také mají tvar podmínky. Jako ukázku můžeme uvést například toto obecné pravidlo, které je určitě důvěrně známé každému, kdo měl alespoň zanedbatelně co do činění s programováním:

```
IF a AND b THEN c
```

Když tuto podmínku ještě více zobecníme, dostaneme vztah:

$$\{situace\, S\} \rightarrow \{akce\, A\}$$

Tento vztah pak můžeme interpretovat tak, že nastala-li v bázi situace S, vykonej akci A.

Levá strana pravidla se nazývá podmínková nebo předpokladová část, antecedent nebo také část vzorů. Strana pravá je nazývána konsekventem nebo důsledkovou

částí. Na levé straně můžeme použít spojky AND a OR, v důsledkové části se může vyskytovat pouze spojka AND:

Pro konkrétnější představu můžeme použít jedno z pravidel báze znalostí popisované aplikace, jehož význam není díky použitým názvům jednotlivých logických členů ani třeba vysvětlovat:

```
IF je_student AND zkouškové_období THEN by_se_hodil_preparát_na_paměť
```

V Prologu bude toto pravidlo vypadat poněkud jinak. Prohodí se totiž jeho levá a pravá strana, takže podmínková část bude vpravo a důsledková nalevo. Obecně pak vypadá tvar pravidla zhruba takto:

důsledek ***jestliže*** *předpoklad*

Zmíněné pravidlo bude tedy postaveno v této posloupnosti:

```
hodil_by_se_preparát_na_paměť :- je_student, má_zkouškové_období.
```

přičemž čárka nahrazuje logickou spojku AND. Pro úplnost dodávám, že spojka OR, tedy logická disjunkce se v Prologu vyjadřuje středníkem. Každé pravidlo nebo fakt je zakončen tečkou.

4.2.3 Predikáty

Prolog je celý postaven na predikátech. Jsou jeho základem. Bylo by tedy vhodné se tomuto termínu před jeho praktickým použitím věnovat trochu i po teoretické stránce. Tento pojem vychází z logiky a v zásadě již byl nepojmenován použit v minulém odstavci. A to jak v obecné, tak v konkrétní formě.

Predikát je jazykový výraz, o jehož obsahu se dá tvrdit, že je buď pravdivý nebo nepravdivý. Predikát může být faktem a může také být konsekventem i antecedentem pravidla. Například `je_student` je predikát a `zkouškové_období` je taktéž predikát. I `hodil_by_se_preparát_na_paměť` je predikát. Predikát může mít argumenty. Počet těchto argumentů se nazývá arita. Název predikátu se pak zapisuje např. `je_student/0`, což znamená, že uvedený predikát nemá žádný argument.

V Prologu není nouze o velkou spoustu dalších pojmů a termínů, z nichž některé se více či méně překrývají. Mám osobní zkušenost, že jejich zavádění mi problém spíše zastřelo, než osvětlilo, když jsem se s Prologem poprvé a možná i podruhé seznamoval. Zkusím tedy vystačit s pojmem predikát, pokud to bude možné. Bude nutné použít i výrazů jako proměnná a konstanta. Ty jsou však natolik obecné a známé, že se v nich určitě neztratíme.

4.3 Příklad použití Prologu v rámci jednoduchého diagnostického systému

Jako příklad podobného typu uvádím opravdu základní diagnostický systém, který jednoduchým způsobem zjišťuje, jestli má daný člověk chřipku, spalničky, zarděnky, příušnice a další nemoci, které jsou vloženy do báze znalostí.

Jedná se o kód právě v jazyce Prolog. Je to velmi jednoduchý příklad, který ale trpí určitými neduhy. Například pokud je zvolena jedna z možností (nebo konkrétněji jeden ze symptomů některé z nemocí), aplikace si nepamatuje, že tato možnost byla zvolena.

Pokud některý z dalších symptomů nevyhoví zadanému dotazu a program nám tím dá najevo, že pacient danou nemocí netrpí, můžeme chtít vědět, jestli třeba netrpí nějakou další, pro kterou jsou v bázi zanesena fakta případně pravidla.

V případě, že další nemoc má některý ze symptomů společný se symptomem nemoci dotazované v předchozí fázi, bude po nás požadována odpověď na stejnou otázku již jednou zodpovězenou, což není moc efektivní.

Například horečka je uvedena jako symptom u většiny nemocí, které máme uložené v bázi znalostí. Jestliže se tedy trefíme až u té poslední, budeme muset odpovídat na otázku ohledně horečky pětkrát. Netýká se to samozřejmě jen horečky, takže pokud by byla báze znalostí řekněme podstatně rozsáhlejší, mohli bychom se dostat do situace, že nám chudák pacient skoná, než tisíckrát odpovíme na to, jestli má horečku.

Tato situace se dá řešit a jelikož i z tohoto jednoduchého programu jsem čerpal inspiraci, budu se tomuto tématu podrobněji věnovat v programové dokumentaci k mojí aplikaci.

I přes jednoduchost tohoto příkladu si však myslím, že je pro názornost zcela dostačující. Na základě zadaných odpovědí dostaneme nějaký více či méně přesný výstup, který nám může sloužit pro orientaci jakou že to nemocí pacient trpí. Problémem však zůstává, že nedostaneme ani náznak toho, jak bychom jej měli začít léčit nebo mu pomoci.

U mého projektu je to přesně naopak. Symptomy a nemoci jsou pouze doplňujícími prvky toho, abychom získali informaci o tom, čím respondentovi býti nápomocni. Rozhodně se nedá mluvit o léčení, ale odborně by se to dalo nazvat například podpora léčby. Ve většině případů se však bude jednat o doporučení produktů, které slouží k předcházení nežádoucích stavů organismu z hlediska zdravotního.

Zde připojuji pro ilustraci krátký popis toho, co uvedený program při hledání pacientovy choroby provádí.

Po spuštění programu v prostředí Prologu, kterému se taktéž budu věnovat ve zvláštním odstavci, začínáme predikátem `go`, který tvoří hlavičku prvního pravidla a spouští samotné dotazování. V těle tohoto pravidla pak pokračujeme vestavěným predikátem `write`, jehož v podstatě vedlejším, nicméně ve většině případů primárně chtěným účinkem je vypsání požadované hlášky na konzoli. V tomto případě je to

otázka na jméno pacienta.

Toto jméno se, po načtení pomocí dalšího predikátu `readln`, v podobě řetězce unifikuje s proměnnou *Patient* a pracuje se s ním pak po celou dobu již jako s konstantou. Na základě predikátu `hypothesis`, kterému je předána již unifikovaná proměnná *Patient* a prozatím volná proměnná *Desease*, kterou se pak bude Prolog snažit unifikovat s některou z uložených nemocí pomocí dalších pravidel.

Hlavičkou pravidla, na které díky posloupnosti predikátů přejdeme, je opět predikát `hypothesis`. V jeho těle pak najdeme již zmíněné symptomy, které se poté budeme pokoušet unifikovat pomocí načítání odpovědí na otázky týkající se projevů nemoci. Vypsání otázky zajistí opět vestavěný predikát `write` a načtení odpovědi predikát `response`, respektive pravidlo jehož je tento predikát hlavičkou. V rámci těla pravidla se pak odpověď načte (`read`) a vypíše (`write`).

Pokud v rámci jedné hypotézy odpovíme na všechny otázky kladně, bude celé pravidlo vyhodnoceno jako splněné a program vypíše informaci o tom, jakou nemocí pacient pravděpodobně trpí.

Jelikož jsou v rámci hypotézy jednotlivé predikáty odděleny čárkami, což v Prologu znamená logický součin, stačí, aby se jeden jediný symptom v rámci hypotézy neshodoval a pravidlo bude vyhodnoceno jako neplatné, což pro nás bude znamenat, že se tato konkrétní hypotéza nepotvrdila.

Prolog pak přejde na následující pravidlo a bude postupovat obdobným způsobem dále, dokud nebude některé pravidlo s uvozujícím predikátem `hypothesis` splněno, čehož výsledek jsem již popsal, nebo dokud neprojde všechny hypotézy uvedené v bázi znalostí.

Jestliže se tak stane a žádná z hypotéz se na základě zadaných symptomů nepotvrdí, bude první pravidlo uvozené predikátem `go` shledáno neplatným a vyhodnocování se přesune na další pravidlo uvedené tím samým predikátem. Úkolem těla tohoto druhého pravidla `go` je vypsání informace o tom, že žádná nemoc uložená v bázi znalostí se neshoduje se zadanými symptomy a program tedy nedokáže vyhodnotit, čím pacient trpí.

Kód popsaného příkladu je pro ilustraci uveden v příloze.

5 Implementace navrženého znalostního systému

5.1 Rozhraní

5.1.1 Rozhraní pro zadávání dat (vyplnění dotazníku)

Jak jsem již předeslal, rozhodoval jsem se při výběru způsobu zadávání mezi třemi možnostmi. Po zvážení všech faktorů, které vstupují do hry jsem se rozhodl tyto způsoby zkombinovat tak, aby zajišťovaly od každého alespoň základ funkčnosti. To znamená, aby se s aplikací dalo pracovat v prostředí jejího budoucího nasazení.

Prvním a důležitým faktorem je samozřejmě doba, kterou bude trvat příprava aplikace pro její prvotní zkušební nasazení. Tato doba by měla být dle mého názoru

co nejkratší. Proto bude v první verzi program používat dotazování přímo v prostředí jazyka Prolog s tím, že se budou pravidly s konsekventem v podobě predikátu `potrebuje(X)` testovat postupně všechny produkty v bázi a průběžně pokládat respondentovi doplňující otázky. Toto je nejrychlejší možnost pro použití v reálném čase, kdy klient bude požadovat doplňující informace potřebné pro jeho rozhodnutí zakoupit si produkt. Podrobněji bude toto popsáno v programové a uživatelské dokumentaci.

Využití stávajícího webového rozhraní dotazníku, kde se výsledná data po vyplnění ukládají ve formě tabulky, která se dá stáhnout v celé řadě formátu, mezi jinými i v csv, je úkol, jehož plné dokončení si kladu do budoucna. Takto stažený soubor se dá použít jako vstup, který přidá fakta do báze znalostí.

Současné webové rozhraní jako takové je sice funkční samo o sobě, protože jsem jej vytvořil primárně pro umožnění zadávání dat elektronickou cestou, což byla alternativa k do té doby pouze tištěné verzi dotazníku. Toto rozhraní má za sebou již relativně dlouhý vývoj, i když se musím přiznat, že velký podíl na něm má společnost Google. Dotazník je totiž postavený na webové aplikaci pro vytváření formulářů právě od této firmy.

Webové rozhraní a otázky v něm pokládané jsou koncipované pro zpracování lékaři. Tato forma doporučování produktů je funkční a je vodítkem pro lidi, kteří chtějí získat prvotní informaci pro podporu rozhodnutí o pořízení konkrétního preparátu. Bohužel však současná formulace otázek a jejich výběr skýtá určitá úskalí. Zmínil jsem se o nich již v úvodních částech mé práce. Proto jsem se rozhodl postupně vytvořit rozhraní nové, které bude postupně rozšiřováno na základě kostry aplikace, kterou v rámci tohoto projektu vytvářím. Toto rozhraní bude k dispozici k nahlédnutí na webové adrese zmíněné v uživatelské dokumentaci a bude jako výstup poskytovat soubory podobné těm uvedeným v ukázkách a přílohách.

Vytvoření samotného dotazníku je záležitost klikání myší a nemá tedy s programováním nic moc společného. Proto se o něm nebudu příliš rozepisovat. Výsledný produkt je vidět na obrázcích 1 – 5. Po vyplnění a stisku tlačítka odeslat jsou data uložena do tabulky, ve které jedna řádka znamená jeden vyplněný dotazník respektive jednoho respondenta. Funkcionalita nově vytvářeného rozhraní bude stejná nebo velmi podobná. Rozdíl se bude ukrývat v pozdějším napojení na Prolog, který bude takto získaná data automaticky vyhodnocovat a posílat je ve formě přílohy e-mailové zprávy správci dat nebo je ukládat do cloudu. V současné době, jak již bylo řečeno, mohu předložit pouze ukázku, jakým způsobem bude Prolog s daty zacházet, pokud budou uložena na počítač správce dat v souboru typu .csv.

Za současného stavu přijde určené osobě, která má na starosti správu dat, e-mailem zpráva, že je na webu respondentem vyplněn nový dotazník. Tato osoba data stáhne jednak za účelem odstranění omylem vyplněných nebo nesmyslných řádků tabulky a v neposlední řadě je také poté odešle lékaři, který provádí vyhodnocení.

Do tabulky se neukládá jméno, pouze kód, který respondent zadává před vyplňo-

váním. Data tedy lékaři dostávají ve zcela anonymní podobě, aby se co nejvíce omezilo nakládání s osobními údaji. Jedinou osobou, která může spárovat konkrétní data s e-mailovou adresou respondenta je správce dat. Ani ten však nemusí znát úplnou totožnost osoby, která vyplněním dotazníku vytvořila záznam v tabulce výsledků. Pro simulaci této činnosti bude nutno využít zmíněné načítání dat ze souboru nebo v nejhorším případě zadávání dat ručně do báze znalostí najednou. Doplňující otázky zde samozřejmě postrádají smysl, protože ve velké většině případů nebudeme mít kontakt s respondentem ve chvíli, kdy budeme data z dotazníku jím vyplněného vyhodnocovat. Je tu samozřejmě možnost zvát si respondenty na vyhodnocení, což z hlediska marketingového také není úplně zavrženíhodná alternativa, nicméně bylo by vhodné mít možnost dostat od potenciálního klienta data a zadat je do systému najednou a očekávat od něho výsledek v podobě doporučených produktů. Tento postup je naznačen při použití predikátu `go`. (více v programové dokumentaci).

Mým aktuálním záměrem je ale i taková funkčnost aplikace, aby její obsluhu, tedy zadávání dat, mohl provádět, po krátkém seznámení se s funkcemi aplikace a s prostředím zvolené implementace Prologu, kterýkoli pracovník tzv. distribučních center, na kterých mají klienti možnost si produkty, které aplikace doporučuje, pořídit. Jsem pevně přesvědčen o tom, že to bude možné a také, že seznámení obstará uživatelská dokumentace k aplikaci, která je součástí této práce, případně nápověda uvedená přímo u výpisů programu při pokládání jednotlivých otázek. Tato nápověda povede obsluhu při zdolávání omezení, která jsou spojena s odpověďmi na jednotlivé otázky, respektive s formátem každé jednotlivé odpovědi a typu dat, která mohou být při té které odpovědi do systému zadávána. .

Klienti vyžadující nadstavbové informace přímo při nákupu produktů jsou tedy druhá skupina, pro které je možno aplikaci využít. Zde naopak můžeme jak vidno otázky klientovi pokládat v reálném čase a sdělovat mu postupné odpovědi tak, jak přicházejí od systému. Tato funkcionalita je naznačena přímým použitím predikátu potrebuje/1, kdy prohledáváme seznam preparátů a program se ptá na data s nimi související aby odvodil jejich vhodnost pro konkrétního klienta, pro kterého se vyhodnocování provádí.

5.1.2 Rozhraní pro vyhodnocování

V plánu je volba rozhraní tak, aby dotazníky bylo možné načítat jednak vcelku, např. přímo z výše zmíněné tabulky uložené ve formátu csv nebo aby bylo možné odpovídat na jednotlivé otázky v prostředí Prologu.

Alternativa přímého zadávání dotazů v Prologu dává osobám určeným k práci s touto aplikací možnost vyplňovat a zároveň vyhodnocovat dotazník v reálném čase. Odstraní se tím nutnost kvůli jednomu dotazníku otevírat prohlížeč, přihlašovat se na portál pro sběr dat a následně stahovat tabulku, jejíž záznam je teprve pak možno použít k vyhodnocení. Tato možnost je tedy vhodnější pro práci „v terénu“ nebo přímo v rámci otevírací doby jednotlivých distribučních center, na kterých se produkty pořizují.

Pokud ovšem respondenti vyplňují dotazník ,,dálkově“ bez toho, aby vešli v přímý kontakt ať už se zmíněným pracovníkem distribučního centra nebo s některým z konzultantů, kteří zajišťují rozšiřování povědomí o produktech, které aplikace testuje, je samozřejmě vhodnější mít k dispozici nástroj, který dokáže data z tabulky přečíst a vyhodnotit několik dotazníků najednou.

I tak je oproti současné situaci změna v podobě ušetření času značná, protože pro každé distribuční centrum je k dispozici ve velké většině případů jeden odborník, který konzultace oficiálně provádí a je na něj uveden kontakt na webových stránkách společnosti. Kdybychom uvažovali, že tomuto odborníkovi bude trvat vyhodnocení dotazníku v průměru 30 minut, a vzali za minimum 5 lidí denně, kteří budou požadovat orientační doporučení produktů, jedná se téměř o tři hodiny práce odborníka za den.

Jelikož zmínění odborníci jsou placeni hlavně za osobní nebo telefonické konzultace, ušetří tato aplikace i ve své rané podobě čas i náklady. Je zde však potřeba zdůraznit, že v současné době není statisticky vyhodnocena míra shody aplikace s názorem odborníka. Tomuto se však hodlám v rámci zkušebního nasazení aplikace věnovat a zvolenou část dotazníků, které budou vyhodnoceny aplikací nechám po stanovené období vyhodnocovat zároveň lékaře (pokud možno stejnou metodou, což znamená procházení preparátů, které budou v bázi znalostí, se snahou zjistit jejich vhodnost pro dotyčného potenciálního klienta).

Rozhraní pro vyhodnocování tak v současné době bude zajišťovat konzolový výstup v prostředí Prologu buď v systému s Windows nebo v systému postaveném na Linuxu. Z Linuxových distribucí jsem vyzkoušel funkčnost na Ubuntu 15.04. Nicméně předpokládám, že aplikace bude fungovat i v ostatních distribucích Linuxu, kde půjde nainstalovat SWI Prolog a jelikož existuje i verze této implementace Prologu pro Mac, nevidím důvod, aby nemohla být aplikace používána i zde. Tuto možnost jsem však neměl příležitost přímo vyzkoušet, jelikož nedisponuji výpočetním prostředkem s tímto systémem.

5.1.3 Budoucí rozšíření rozhraní

Jak vyplývá z výše uvedeného, existuje spousta možných rozšíření rozhraní pro koncové uživatele aplikace. Kromě možnosti automatického načítání z tabulky je zde ještě jedna možnost k zamyšlení do budoucna. Stále se totiž potkáváme s dotazníky v tištěné formě. Nabízí se tedy varianta vstupu ve formě strojového čtení naskenovaného obrazu takto vyplněného formuláře. Otázkou zůstává, jestli množství tištěných dotazníků, které se s postupující dobou stále snižuje, stojí za úsilí, čas a další náklady spojené s rozšířením aplikace tímto směrem.

5.2 SWI Prolog

Pro zkušební nasazení aplikace do praxe je v jejím současném stavu nutné použít některou z implementací Prologu. Zvolil jsem SWI Prolog. Jednak má velmi příjemné

prostředí, které se pomocí rozšíření pro IDE Eclipse dá provozovat s využitím většiny komfortních služeb, které Eclipse nabízí. Ať už je to, mimo jiné, automatické načtení (re-consult) souboru po jeho uložení, automatické přechody z debug módu do módu trace při nalezení breakpointu a současná implementace grafického debuggeru procházejícího přímo zdrojový kód programu po řádcích tak, že je vidět kde se zrovna v určité chvíli v programu nacházíme, což při použití klasického ,,trasování“ predikátem trace není přímo v prostředí Prologu úplně možné. Dále se mohu zmínit i o možnosti napojení na systémy správy verzí jako je např. Git.

Další argument, který pro tuto implementaci hovoří, je její cena pro komerční použití. Je totiž zdarma a dle informací přímo z webu SWI Prologu je toto využití dokonce vítáno, jelikož přináší rychlejší odhalování chyb v implementaci a tím pádem urychluje samotný vývoj této implementace.

V neposlední řadě umí tato implementace pracovat se soubory, v nichž jsou hodnoty uloženy ve formě oddělené čárkami nebo tabulátory. Tedy csv a tsv. To je pro budoucí vývoj aplikace velmi žádoucí.

6 Programová dokumentace

6.1 Úvod

Programová dokumentace je rozdělena do dvou částí. Jednak je představena většina predikátů s jejich ukázkami a popisem jejich funkce. Jelikož některé z nich jsou několik desítek řádek dlouhé, uvádím zde vždy hlavičku a několik řádek těla. Kompletní kód je na přiloženém médiu. Seznam neobsahuje některé pomocné predikáty sloužící pouze při ladění aplikace. Jejich činnost bude případně uvedena v popisu funkcí aplikace z hlediska pravidel, faktů a jejich vzájemného propojení. Z toho vyplývá, že tento popis aplikace tvoří druhý blok programové dokumentace.

6.2 Seznam predikátů

6.2.1 Predikát `dynamic/1`

informuje interpret, že predikáty, jejichž seznam následuje se mohou měnit za běhu programu pomocí predikátů `assert/1` a `retract/1`.

```
:- dynamic
        casta_chripka/0,
        gender/1,
        datum/3,
        vek/1,
```

6.2.2 Predikáty `file_in a file_out`

specifikují cestu ke vstupnímu a výstupnímu souboru, aby se nemusela vypisovat do každého predikátu, kde bude potřeba.

```
file_in('c:/users/max/git/bp/bp/example.csv').
file_out('c:/users/max/git/bp/bp/outfile.txt').
```

6.2.3 Predikát `read_data/0` a `read_data_dalsi/0`

provádí v případě prvního načtení dat ze souboru csv a odstranění první řádky, která obsahuje hlavičky sloupců a není tudíž žádoucí ji načítat do příslušných struktur. Použije se pro první řádek platných dat. V případě druhého predikátu je provedeno odstranění řádky minulé a načtení řádky další(pokud existuje), která obsahuje data naimportovaná ze souboru `csv` do báze znalostí jež jsou bezprostředně po načtení souboru definována predikátem `row/12`. Tento predikát se používá pro druhý a každý další případný řádek platných dat.

```
read_data :-
        file_in(File),
        csv_read_file(File,Rows),
        maplist(assert, Rows),
        retract(row(_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)),!,
        get_gender(G),
        asserta(known(_,gender,G)),
```

```
read_data_dalsi :-
        retract(row(_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)),!,
        get_gender(G),
        asserta(known(_,gender,G)),
```

6.2.4 Predikát `vymaz/0`

zajišťuje odstranění dynamicky vložených dat pomocí predikátu `asserta/1` do báze znalostí. Jedná se konkrétně o predikáty `known/3` a `row/12`.

```
vymaz :-
        retractall(known(_,_,_)),
        retractall(row(_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)).
```

6.2.5 Predikáty `get_xy`

zabezpečují načtení požadovaných dat do patřičných struktur z příslušného místa aktuální řádky predikátu `row/12`.

```
get_gender(G):-
        row(_,_,G,_,_,_,_,_,_,_,_,_,_,_,_,_),!.
get_form_code(Code):-
        row(_,Code,_,_,_,_,_,_,_,_,_,_,_,_,_,_),!.
get_curr_date(Y,M,D):-
        row(Datum,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_),!,
```

6.2.6 Predikát `go/0`

byl vytvořen za účelem vyžádání všech dat od uživatele najednou a vyhodnocení prvního preparátu, který je na řadě, tzn. prvního predikátu `potrebuje/1`.

```
go :-
        zadani_kodu_ruc,
        gender,
        age,
        obdobi,
        miry,
```

6.2.7 Predikát `go_file/0`

načítá data z csv souboru a připravuje jejich rozvrstvení do příslušných programových struktur.

```
go_file :-
        read_data,
        vyhodnoceni,!.
```

6.2.8 Predikát `zadani_kodu_ruc/0`

Každý dotazník má svůj identifikační kód, pokud je vyplňován přes webové rozhraní. Pokud jej vyplňujeme ručně, je nutno zadat nějakou kombinaci znaků, podle které budeme pak konkrétní výstup ze souvisejících vstupních dat hledat ve výstupním souboru, např. pokud bude vyplňováno více dotazníků. Tento predikát nám zajistí získání identifikačního kódu od uživatele. Zapíše také tento kód na konzoli i do výstupního souboru.

```
zadani_kodu_ruc :-
        ask('Zadejte kod dotazniku (napr. D01-RR-MM-DD)', code, Code),
        known(_, code, Code),
        write(Code),
        nl,
```

6.2.9 Predikáty `vyhodnoceni/0` a `vyhodnoceni_user/0`

slouží k samotnému vyhodnocení zadaných vstupních dat probíráním jednotlivých predikátů potrebuje/1 a tím pádem zkoušením vhodnosti příslušných preparátů, které tyto predikáty zastupují, z hlediska vhodnosti pro respondenta. První z nich je použit při vyhodnocení dat načtených ze souboru a druhý při vyhodnocování na základě ručního zadávání dat.

```
vyhodnoceni :-
        known(_,code,Code),
        write(Code),
        nl,
```

6.2.10 Predikáty `konec/0` a `konec_user/0`

provádějí oba interakci s uživatelem ohledně pokračování v zadávání dalších dat, potažmo vyhodnocování dalších dotazníků nebo případného ukončení a vymazání dynamicky vložených dat predikátem `retractall/1`.

```
konec :-
        write('Dalsi dotaznik/Konec - d/k? '),
        read_string(A),
        A = d,
        pokracuj;
        write('Koncim a mazu dynamicky vlozena data.'),
        retractall(known(_,_,_)),!,
        retractall(row(_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)),!.
```

6.2.11 Predikáty `pokracuj/0` a `pokracuj_user/0`

dovolí uživateli načíst nová data pro vyhodnocení dalšího dotazníku. Mimo jiné také provedou odřádkování ve výstupním souboru.

```
pokracuj :-
        retractall(known(_,_,_)),
        file_out(File),
        open(File,append,S),
        nl(S),
```

6.2.12 Predikáty `read_string/1` a `read_string_file/1`

načítají vstupní data ve formě řetězce a kromě toho, že umožňují v prostředí Prologu zadávat uživateli data bez závěrečné tečky vyřešil tento postup problém nefunkčnosti programu ve Windows v případě, že byly načítány různé typy dat (např. v jednom predikátu celé číslo a v následujícím pak řetězec). V Linuxu aplikace fungovala bez problému, kdežto ve Windows se do vstupu typu řetězec načítal vstup složený z nesmyslných znaků. Nepodařilo se mi zatím zjistit příčinu.

```
read_string(String) :-
        read_line_to_codes(user_input, Codes),
        atom_codes(String, Codes).
```

6.2.13 Predikát `multivalued/1`

má za úkol odlišit predikáty, které jsou typu ano/ne od těch, které mohou nabývat více různých hodnot. Tento predikát pak figuruje v rámci kontroly, zda požadovaná hodnota pro výpočet nebo vyhodnocení již byla od uživatele získána nebo je třeba jej o ni požádat.

6.2.14 Predikát `age/0`

tvoří konsekvent pravidla pro kontrolu nebo zadání dat pro výpočet věku. Je to vlastně rozcestník pro predikát `age/3`, který pak provádí část výpočtu věku. Konkrétně tu část, kdy jsou si rovny měsíce aktuálního data a data narození. Opět se spoléhá na vyhledávání požadované informace za pomoci predikátu `ask_auto/3`.

```
age :-
        ask_auto(_, vek, _),!;
        bdate(BY,BM,BD),
        curr_date(Y,M,D),
        vypocet_veku(BY,BM,BD,Y,M,D).
```

6.2.15 Predikát `curr_date/3`

je použit k získání aktuálního data dle data systémového. Používá dotaz za pomoci predikátu `ask_auto/3` a vestavěné predikáty SWI Prologu pro zpracování časových údajů `get_time/1` a `stamp_date_time/3`.

```
curr_date(Y,M,D) :-
        ask_auto(_,curr_date(Y,M,D),_),!;
        get_time(TS),
        stamp_date_time(TS,date(Y,M,D,_,_,_,_,_,_),'local').
```

6.2.16 Predikát `bdate/3`

je vyhrazen pro interakci s uživatelem za účelem získání jeho data narození. Obdržený údaj je zpracován vestavěným predikátem SWI Prologu `parse_time/2`.

```
bdate(Y,M,D) :-
        ask_auto(_,bdate(Y,M,D),_),!;
        write('Zadej datum narozeni: '),
        read_string(Datum),
        parse_time(Datum, Stamp),
        stamp_date_time(Stamp,date(Y,M,D,_,_,_,_,_,_),'local').
```

6.2.17 Predikáty `obdobi/0` a `obdobi/2`

zjišťují na základě aktuálního systémového data, zda právě probíhá zkoumané období. Pokud je již datum zadán, použije jej pokud je známo (uloženo v bázi znalostí pomocí predikátu assert/1), že zadané období buď probíhá nebo neprobíhá, je použita tato informace bez nutnosti zkoumat hlouběji.

Pro ilustraci jsou zde pouze tři období a ještě predikát `zkouskove_obdobi/0`, založený na podobném principu, která hrají roli při zjišťování vhodnosti některého z preparátů vyhodnocovaných pravidlem `potrebuje/1`. Jedná se konkrétně o jaro, podzim a období, kdy v našich zeměpisných šířkách je vysoká pravděpodobnost, že

je třeba se chránit před spálením od slunce respektive spíše od toho být po pobytu na slunci nepříjemně narůžovělý.

```
obdobi :-
        obdobi(jaro, new_value),
        obdobi(slunce_opaluje, new_value),
        obdobi(podzim, new_value).
```

6.2.18 Predikáty `get_value/3` a `get_value_from_user/2`

prvně jmenované mají za úkol zjistit hodnoty pro predikáty obsažené v jejich argumentech, většinou na základě výpočtu pomocí predikátů `vypocet_xy/?`. Ty druhé pak zabezpečí získání příslušných dat od uživatele, pokud tato nejsou nalezena v aktuální bázi znalostí.

```
get_value(obdobi(jaro), X, V) :-
        ask_auto(_, datum_m, _),
        vypocet_jaro(V,X).
```

```
get_value_from_user(je_unavenej, X):-
        ask('Byvate casto unaveni? (a-ano,n-ne) ', je_unavenej, X).
```

6.2.19 Predikát `vypocet_xy/?`

je množina predikátů, které vypočítávají hodnoty, jež jsou pak použity predikáty `get_value/3`. Například se vypočítá na základě zadané výšky a váhy BMI index a na základě tohoto indexu pak určí, zda má respondent nadváhu nebo je v normálu. Nebo v případě zjišťování aktuálního období, které má vliv na doporučení nějakého produktu se dle měsíce aktuálního data zjistí, zda se aktuálně pohybujeme v časovém intervalu stanoveném pro toto období. Vypočítává se taktéž věk ze zadaného data narození. Postup je takový, že se vyhodnocuje nejdříve rok, kdy se odečítá rok narození od roku aktuálního, a pak se pokračuje na vyhodnocení měsíce. V případě, že jsou měsíce rozdílné, výpočet končí. Pokud však měsíc aktuálního data souhlasí s měsícem narození, pokračuje se dál na vyhodnocení v granularitě jednotlivých dnů. Jestliže jsou dny rozdílné, provede se odečet, pokud souhlasí, aplikace, pokud je v módu pro vyhodnocování v reálném čase, popřeje respondentovi k narozeninám. To je také jeden z důvodů, proč jsem se rozhodl ptát se uživatelů na datum narození a ne přímo na věk, který mi o tom, kdy mají narozeniny nic neřekne.

V ukázce je uveden výpočet BMI indexu a jeho vyhodnocení. Výpočet je zde uveden bez zbytečných nuancí a jsou k dispozici pouze dvě hladiny (nadváha a normální váha). V budoucích verzích bude úrovní přibývat a výpočet bude zpřesněn.

```
vypocet_tlustosti(Vyska_s, Vaha_s, _, V) :-
        atom_number(Vyska_s,Vyska),
        atom_number(Vaha_s,Vaha),
        Vaha/(Vyska/100)**2 >= 30.0 ->
```

6.2.20 Predikát `gender/0`

zkontroluje, zda je v bázi znalostí již uložena informace o pohlaví respondenta a pokud tomu tak není, vyžádá si zadání této informace.

```
gender :-
        ask('Zadejte pohlavi (m-muz, w-zena) ', gender, new_value).
```

6.2.21 Predikát `vek/1`

zkontroluje, zda je v bázi znalostí uložena informace o věku respondenta. Pokud ne, zjistí, zda je uloženo alespoň datum respondentova narození. V případě neúspěšného hledání tohoto údaje si jej následně pomocí pravidla s konsekventem `age/0` vyžádá.

```
vek(X) :-
        ask_auto(_, vek, X),!;
        age.
```

6.2.22 Predikáty `potrebuje/1`

jsou konsekventem pravidel, která tvoří nosnou část aplikace. Díky těmto predikátům se dle dat vyhodnocuje vhodnost jednotlivých preparátů, které jsou zadány jako argumenty těchto predikátů. V rámci těla těchto predikátů najdeme pomocná pravidla, která buď zjistí hodnotu v bázi znalostí nebo, v případě vyhodnocování v reálném čase, data přímo od respondenta.

```
potrebuje(pycnogenol) :-
        je_unavenej,
        write('Poridte si Pycnogenol, abyste nebyli unaveni. '),
        file_out(File),
        open(File,append,S),
        write(S,'Poridte si Pycnogenol, abyste nebyli unaveni.'),
        nl(S),
        close(S),
        nl, fail.
```

6.2.23 Predikát `miry/0`

slouží pro získání výšky a váhy od uživatele. Pokud jsou již tyto hodnoty známy (přidány predikátem assert/1 do báze znalostí), použijí se tyto bez dalšího dotazování respondenta.

```
miry :-
        ask_auto(_, vyska, _),
        ask_auto(_, vaha, _),!;
        write('Zadej vysku: '),
        read_string(Vyska),
```

6.2.24 Predikát `je_tlustej/1`

vyhodnocuje BMI index respondenta na základě jeho výšky a váhy. Pokud je již hodnota o nadváze nebo normální váze v bázi znalostí, použije tuto. Pokud není, použije případně již známou výšku a váhu, nebo v případě, že nenalezne ani tuto informaci, požádá uživatele, aby výšku a váhu zadal.

```
je_tlustej(a) :-
        ask_auto(_, je_tlustej, a),!.
```

6.2.25 Predikát `ma_pred_sebou_intenzivni_obdobi/0`

je pravdivý v případech, kdy má úspěch pravidlo s konsekventem `je_student/0` a současně `zkouskove_obdobi/0` nebo pravidlo s konsekventem `je_tehotna/0`. Vyjadřuje potřebu respondenta adaptovat organismus na vyšší zátěž, respektive vede aplikaci k doporučení preparátů, které jsou k tomu určené.

```
ma_pred_sebou_intenzivni_obdobi :-
        je_student,
        zkouskove_obdobi,!.
```

6.2.26 Predikát `unava/0` a `je_unavenej/1`

zkoumá, zda je období, ve kterém se obecně projevuje u lidí více únava nebo jestli případně v jiném období odpoví kladně na otázku, zda únavu dle svého názoru ve zvýšené míře pociťují. Predikát unava/0 sdružuje všechny možnosti, kdy bude o respondentovi platit, že by mohl být unavený a je koncipován tak, že jeho unifikace bude vždy úspěšná.

6.2.27 Predikát `je_treba_se_namazat/0`

je považován za pravdivý v případě, že je období, kdy na našem území slunce produkuje zvýšenou míru UV záření nebo pokud respondent odpoví kladně na otázku, že v dohledné době plánuje cestu do oblastí, kde tomu tak bude.

```
je_treba_se_namazat :-
        ask_auto(_, obdobi(slunce_opaluje), a),!.
```

6.2.28 Predikát `do_teplych_kraju/1`

souvisí s predikátem předchozím a zjišťuje, zda uživatel odpověděl tak či onak na otázku ohledně plánované cesty do tropických oblastí nebo obecně oblastí, ve kterých je v tu dobu nutné se chránit před sluncem.

```
do_teplych_kraju(X) :-
        ask('Jedete v dohledne dobe na dovolenou za sluncem?',
         do_teplych_kraju(X), X).
```

6.2.29 Predikát `padaji_vlasy/0`

je pravdivý pokud respondent odpoví kladně na otázku, zda mu vypadávají ve zvýšené míře vlasy.

```
padaji_vlasy :-
        ask_auto(_, padaji_vlasy, a).
```

6.2.30 Predikát `tehotnost/0` a `je_tehotna/0`

zjišťuje, zda respondent je žena a pokládá otázku ohledně její gravidity. Predikát `tehotnost/0` opět sdružuje všechny predikáty, u kterých hraje gravidita roli a vždy projde (je úspěšně unifikován), protože je na tom závislé úspěšné provedení predikátu `go/0` pro získání všech dat, která jsou reprezentována tělem tohoto pravidla.

```
tehotnost :-
        je_tehotna,!;
        !.
```

6.2.31 Predikát `je_student/0`

zjišťuje, zda respondent studuje.

```
je_student :-
        ask('Studujete? ', je_student, a).
```

6.2.32 Predikát `casta_chripka/1`

zeptá se uživatele, zda má často chřipku, respektive zkontroluje přítomnost tohoto faktu v bázi znalostí.

```
casta_chripka :-
        ask('Mate casto chripku? ', casta_chripka, a).
```

6.2.33 Predikát `imunita/0`

zjišťuje buď podle období nebo dotazem na častou nemocnost, zda respondent potřebuje podpořit imunitu. Pokud je informace již zanesena v bázi znalostí, použije ji, pokud ne provede příslušné dotazy.

```
imunita :-
        obdobi(podzim,a),
        write('Poridte si Bi-iomare a zustante i na podzim a v zime'),
        write(' odolni proti nemocem. '),
```

6.2.34 Predikát `vyzkouset_novy_intim/0`

ptá se uživatele na ochotu vyzkoušet nový produkt v oblasti intimní hygieny v případě, že je jím žena.

```
vyzkouset_novy_intim :-
        ask('Chcete vyzkouset novy pripravek pro
          intimni hygienu? (ano/ne) ', vyzkouset_novy_intim, a).
```

6.2.35 Predikát `intenzivni/0`

sdružuje případy, kdy bude u respondenta zvýšená pravděpodobnost, že bude potřebovat posílit organismus proti únavě a snížit míru opotřebení volnými radikály. Spolupracujícími predikáty jsou v tomto případě `ma_pred_sebou_intenzivni_obdobi/0` a `je_unavenej/0`.

```
intenzivni :-
        ma_pred_sebou_intenzivni_obdobi,
        !.
```

6.2.36 Predikát `na_hlavu/0`

pokrývá případy, kdy respondent splňuje předpoklady k doporučení produktu na podporu mozkové činnosti. V prvním případě pomocí predikátu `je_student/0` ověří, zda dotyčný studuje, buď najde informaci přímo v bázi znalostí nebo si ji vyžádá položením doplňující otázky, a v případě druhém zkontroluje respondentův věk a pokud tento je vyšší než 70 let, pravidlo je splněno. Díky tomu je pak úspěšně unifikován i příslušný predikát `potrebuje/0`.

```
na_hlavu :-
        vek(X),
        known(_,vek,X),
        X > 70,
        write('Poridte si Fosfosercaps pro snizeni '),
        write('rizika onemocneni Alzheimerovou chorobou. '),
```

6.2.37 Predikáty `ask/3`, `ask_auto/3` a `known/3`

zabezpečují kontrolu již uložených faktů do báze znalostí a v případě potřeby vyvolají interakci s uživatelem nebo zjistí potřebnou informaci z dostupných zdrojů. Rozdíl mezi nimi je v tom, že predikát `ask/3` se ptá respondenta a naproti tomu `ask_auto/3` systému. V některých případech jsou tyto dva predikáty kombinovány dohromady. Do budoucna plánuji sjednotit postup aplikace tak, aby predikáty striktně procházeli buď ask/3 v případě uživatelských dat a ask_auto/3 v případě dat pocházejících ze systému, např. aktuální datum.

Predikáty jsou koncipovány tak, že postupně kontrolují přítomnost faktů a pravidel v bázi znalostí. Postup je několikafázový. Nejdříve se kontrolují všechny argumenty, což znamená, že nám jde o otázku, název hodnoceného predikátu i odpověď. Pak se kontroluje přítomnost hodnoceného predikátu bez ohledu na odpověď a v případě, že odpověď na otázku, kterou vyhodnocované pravidlo reprezentuje, je typu ano/ne, zjišťuje se také, zda je odpověď, která pochází od zdrojového predikátu potrebuje/0, zastupujícího vlastně vyhodnocovaný produkt, nebo predikátu pomocného, jako třeba je_unavenej/0, je shodná s odpovědí zadanou uživatelem nebo již uloženou v bázi znalostí(proměnné A a V).

Predikát `known/3` je pak konsekventem speciálního pravidla, které se používá v rámci predikátů `ask/3` a `ask_auto/3` pro zanesení predikátem `assert/1` a následnou kontrolu přítomnosti v bázi znalostí při procházení pravidel vyhodnocujících potřebu jednotlivých doporučovaných produktů.

```
ask(Q, P, A):- % ziskava hodnoty a prirazuje k predikatum
        write(Q),
        \+ multivalued(P),
        read_string(V),
        asserta(known(Q,P,V)),!,
        evaluate(P,A,V);
```

```
ask_auto(_, P, A):-
        \+ multivalued(P),
        get_value(P, A, V),
        asserta(known(_, P, V)),!,
        evaluate(P,A,V);
        get_value(P, A, V),
        asserta(known(_, P, V)).
```

6.3 Popis činnosti aplikace

6.3.1 Načtení dat ze souboru

Po zadání dat do dotazníku přes webové rozhraní se odpovědi na jednotlivé otázky uloží do tabulky. Tabulka se dá stáhnout jako soubor s hodnotami oddělenými čárkou. Tento soubor bude mít za současného stavu zmíněného rozhraní formát zhruba takový, jaký je vidět v následující ukázce. Soubor obsahuje v prvním řádku hlavičky sloupců, které jsou před použitím vlastních dat odstraněny. Zde konkrétně máme dva řádky platných dat.

```
datum_vyp,code,gender,b_date,vyska,vaha,unava,...,intim
2015-06-21,XCV-09-JK,m,1972-11-24,180,100,a,,n,a,n,
2015-06-21,FHF-77-IU,w,1972-02-25,172,57,n,a,a,a,a,n
```

Načítání a vyhodnocování spustíme zadáním dotazu `go_file/0`. Prolog najde příslušné pravidlo se stejnojmenným konsekventem v bázi znalostí a bude se po-

koušet o unifikaci. V těle pravidla (viz kód v příloze) je jako první na řadě predikát `read_data/0`, který opět tvoří hlavičku dalšího pravidla a tedy vyvolá další pokus o unifikaci a přejde ke zpracování těla, ve kterém se mimo jiné nachází i vestavěný predikát SWI Prologu `csv_read_file/2`, který se pokusí načíst soubor, obsažený v jeho prvním argumentu. Druhý argument reprezentuje výstup, se kterým bude naloženo pomocí `maplist/2` tak, že vloží pomocí svého prvního argumentu, což je vestavěný predikát `assert` do báze znalostí seznam řádek obsahujících data ze souboru. Bude k tomu použit predikát `row/?`, který bude mít aritu podle počtu hodnot v řádce. V našem případě to tedy bude 12.

Tyto hodnoty jsou pak z aktuálního řádku jedna po druhé odpovídajícími predikáty `get_xy/?` vloženy do příslušných struktur definovaných predikátem known/3.

Po vložení dat může začít vlastní vyhodnocování. To začne načtením kódu dotazníku, jeho vypsáním do konzole prostředí SWI Prologu a také do výstupního souboru. Pak začne procházení jednotlivých preparátů určených příslušnými pravidly uvozenými predikátem potrebuje/1. Všechna pravidla s konsekventem potrebuje/1 jsou koncipována tak, aby vždycky selhala a aplikace tak bez čekání přešla k vyhodnocení dalšího produktu, dokud neprojde všechny uložené v bázi znalostí.

Po tomto kroku následuje odřádkování na konzoli a unifikace predikátu `konec/0`, který s přispěním predikátu `pokracuj/0` dá uživateli na výběr, zda pokračovat vyhodnocením další řádky souboru nebo skončit a vymazat dynamicky vložená data. Při volbě pokračování ve vyhodnocování je unifikován predikát `read_data_dalsi` a postup je dál vcelku stejný od načtení dat po průchod jednotlivými pravidly `potrebuje/1`. Jestliže si uživatel zvolí konec, aplikace vymaže data vložená predikáty assert a skončí.

6.3.2 Manuální načítání dat před vyhodnocením

Pro tento způsob získání dat k vyhodnocení použijeme predikát `go/0`. Systém se nás postupně ptá na potřebná data a poskytuje nám nápovědu k formátu, v jakém mají být data formulující odpověď k související otázce zadána. Jako první údaj je nutno vyplnit identifikační kód dotazníku, jehož formát je sice doporučen v nápovědě k otázce, ale dodržet jej není nutno. Kód by měl sloužit k tomu, aby se daly ve výstupním souboru výstupy z jednotlivých dotazníků co nejsnadněji odlišit.

Po získání všech potřebných dat dochází opět k vyhodnocení, reprezentovaném predikátem `vyhodnoceni_user/0`. Vyhodnocení se nijak dramaticky neliší od původního případu, kdy jsou data načtena ze souboru. Je k tomu použit téměř totožný postup a úplně totožné predikáty `potrebuje/1`. Jediný rozdíl spočívá v tom, že i při vyhodnocování jsou uživateli pokládány doplňující otázky pro ukázku, jak by mělo fungovat vyhodnocování založené od začátku na průběžném kladení otázek(viz další odstavec 6.3.3).

Po vyhodnocení je zajištěn výběr možnosti pokračovat vyhodnocením dalších dat, která budou zadávána opět ručně, nebo skončit a vymazat data doplněná do báze znalostí při aktuálním vyhodnocování. Tyto volby jsou řešeny téměř totožně

jako u předchozího typu načítání dat. Jsou však pro ně vyhrazeny vlastní predikáty, které zohledňují mírné odlišnosti při tomto postupu. Jedná se o `konec_user/0` a `pokracuj_user/0`.

6.3.3 Načítání dat průběžně s okamžitým vyhodnocováním

Tato podoba načítání dat není v této verzi ještě plně implementována. Měla by sice fungovat, ale její funkčnost není dostatečně otestována. Je to volba, která je vhodná k použití při získávání nových klientů nebo při servisu klientů současných.

Tato forma zadávání dat nevyužívá žádného s predikátů go_xy/0. Vyhodnocení začneme zadáním dotazu potrebuje(X) a systém pak u každého z produktů hledá příslušná data a pokud je nenajde, vyžádá si je od uživatele. Pro ukázku této funkčnosti jsem zkombinoval položení několika doplňujících otázek s typem načítání dat uvedeným v minulém odstavci.

7 Uživatelská dokumentace

7.1 Práce s webovým rozhraním

Webové rozhraní pro zadávání dat do dotazníku najdete po zadání této adresy do adresního pole vašeho internetového prohlížeče:

`https://docs.google.com/forms/d/1KEt757Uon4cX4RGvAhZJwzur_C62KLHVjebS33o2714/viewform`

Dále pak postupně vyplňte všechna data dle nápovědy a stiskněte tlačítko Odeslat ve spodní části formuláře. Vyplněné výsledky pak najdete v tabulce.

7.2 Práce s textovým prostředím jazyka Prolog (SWI Prolog)

7.2.1 Instalace prostředí SWI Prolog

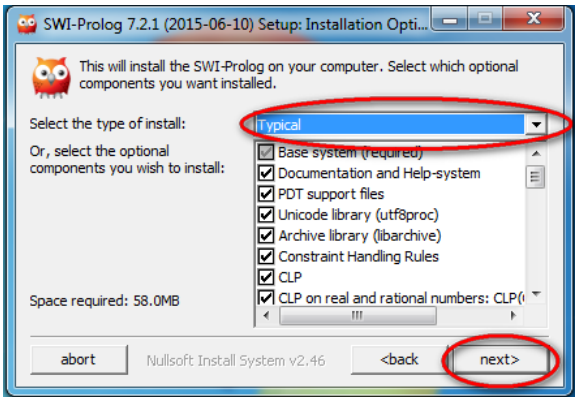
Windows

V návodu a na snímcích obrazovky jsou použita Windows 7. Pro ostatní verze je však postup obdobný. SWI Prolog je možno stáhnout na této adrese:

`http://www.swi-prolog.org/download/stable`.

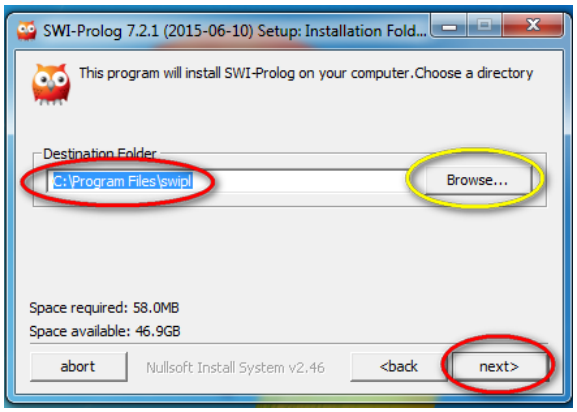
Po stažení spusťte instalační soubor a postupujte dle návodu níže.

Na základě spuštění instalačního souboru by se mělo objevit stejné nebo hodně podobné okno, jako je na obrázku 8. Ponechte *typ instalace (type of install)* na hodnotě *Typical* a klikněte na tlačítko *next*.

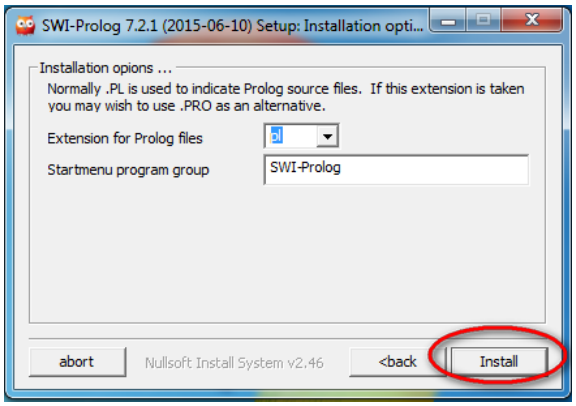


Obrázek 8: Instalace SWI Prologu WIN - fáze 1

V dalším dialogovém okně (viz obrázek 9) pak zkontrolujte složku, do které se bude SWI Prolog instalovat a pokud vám vyhovuje, stiskněte tlačítko *next*. V opačném případě tlačítkem *Browse* vyhledejte složku, do které budete chtít tuto implementaci Prologu nainstalovat a teprve poté stiskněte tlačítko *next*.



Obrázek 9: Instalace SWI Prologu WIN - fáze 2

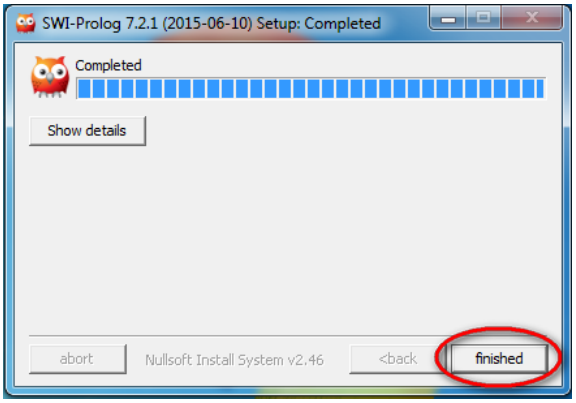


Obrázek 10: Instalace SWI Prologu WIN - fáze 3

Poté nám instalační program nabídne zvolit si příponu souborů, které se s Prologem asociují (Extension for Prolog files) a také název položky v nabídce Start

ve Windows. Doporučuji ponechat výchozí hodnoty a stisknout tlačítko *Install* (viz obrázek 10).

Pokud vše v rámci instalace proběhne v pořádku, měli byste vidět závěrečné okno (viz obrázek 11), ve kterém klikněte na tlačítko *finished*.

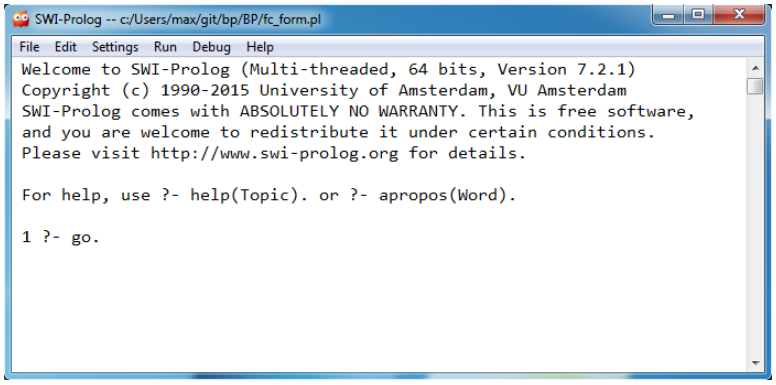


Obrázek 11: Instalace SWI Prologu WIN - fáze 4

Linux
Instalace na Linuxu je vcelku jednoduchá. SWI Prolog se nachází v repozitářích mnoha distribucí. V našem příkladě budeme používat Ubuntu 15.04.

7.2.2 Zadávání dat pod Windows

Pro spuštění prostředí SWI Prologu klikněte postupně na *Start→Všechny programy→SWI Prolog→Prolog*. Otevře se okno Prologu, jak je vidět na obrázku 12.



Obrázek 12: Spuštění SWI Prologu

Před použitím doporučuji nastavit font na consolas nebo jiný, rozumně čitelný font, což lze lehce provést vybráním položky menu *Settings* a pak *font*. Dále z nabídky vybereme *consolas* nebo jiný váš oblíbený font a potvrdíme. Prostředí Prologu je pak podstatně čitelnější. Je možné, že to závisí na monitoru nebo možná i kvalitě zraku, ale zkoušel jsem SWI Prolog pod Windows na třech monitorech a na všech vypadal výchozí font strašlivě.

Načíst soubor je pro uživatele aplikace (neprogramátora) nejpřirozenější pomocí menu, konkrétně nabídky *File→Consult*. Provedení této akce nám umožní vybrat soubor, který chceme načíst. V tomto případě to bude soubor uložený na přiloženém CD s názvem `fc_form.pl`.

Ještě snadnější formou je typický dvojklik na soubor v souborovém manažeru (například průzkumník Windows). Tato akce způsobí jednak otevření okna, které je vidět na obrázku 12 a současně také načtení souboru. Můžeme tedy začít rovnou vyhodnocovat dat.

Pokud chceme vyhodnocovat s ručním zadáváním dat, napíšeme do okna Prologu dotaz *go*. Dotaz musí být ukončen vždy tečkou. Zadávání konkrétních dat je pak již bez tečky!

Konkrétně budeme psát za otazník a pomlčku, jak je vidět na obrázku 12. Jako první se nás systém zeptá na identifikační kód dotazníku, který si můžeme zvolit v podstatě libovolně. Doporučený formát kódu je uveden v nápovědě přímo za výzvou k zadání kódu v okně Prologu.

V tomto případě zadáme většinu dat najednou před tím, než se začnou vyhodnocovat. Pokud systém má k dispozici dat dostatečné množství, začne vyhodnocovat a pokládat nám doplňující otázky, na které opět odpovídáme jejich vepsáním v požadovaném formátu do okna SWI Prologu. Formát dat, který je nutno při zadávání do systému použít je vždy definován na konci položené otázky, např. „Zadejte pohlavi: (m-muz, z-zena)“, z čehož vyplývá, že zapíšeme malé m, pokud jde o muže a malé z pokud se jedná o ženu.

Pokud odpovíme všechny doplňující otázky budeme mít v okně Prologu odpovědi v podobě doporučení vhodných preparátů, samozřejmě proložené doplňujícími otázkami a odpověďmi na ně. Ve výstupním souboru budou pouze odpovědi pod příslušným kódem.

Obsah výstupního souboru může vypadat například takto:

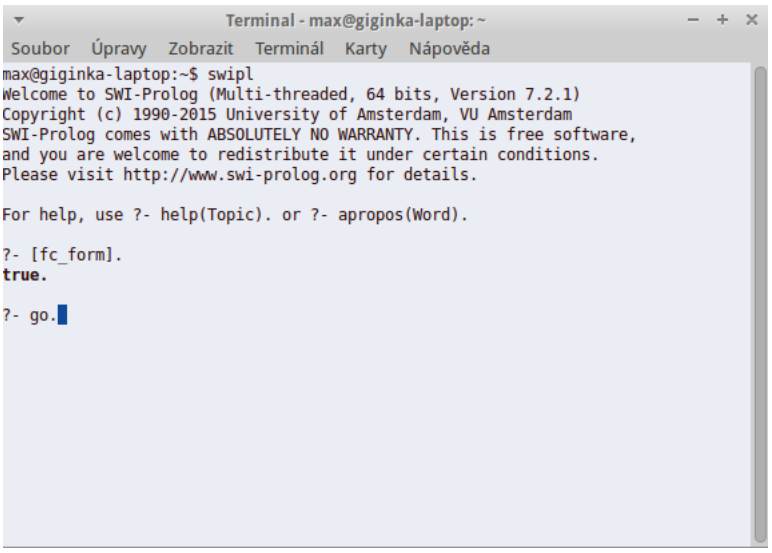
```
D01
Poridte si Betacaps pro snizeni rizika spaleni na slunci.



D02
Vas BMI index indikuje nadvahu. Poridte si Slimcaps.
Poridte si Pycnogenol, abyste nebyli unaveni.
Poridte si intimni gel s extraktem z dubove kury.
Poridte si Evocaps, at vam nepadaji vlasy.
Poridte se Bi-iomare a zvyste svoji imunitu.
Poridte si Betacaps pro snizeni rizika spaleni na slunci.
Poridte si Dynaforce a zvladnete toho vice.
Poridte si Fosfosercaps a uceni vam pujde lepe do hlavy.
V tehotenstvi je dobre doplnit zelezo. Poridte si Ferrofortis.
```

Máme-li k dispozici soubor z tabulky s výsledky získanými vyplněním dotazníků přes webové rozhraní, postupujeme mírně odlišným způsobem. V první řadě zajistíme, aby tento soubor, který musí mít koncovku `*.csv` byl ve stejném adresáři (složce) jako soubor aplikace tedy fc_form.pl. Pokud jsme úspěšně absolvovali krok kopírování vstupního souboru do správné složky, napíšeme do okna Prologu dotaz *go_file* a opět zakončíme tečkou. Jestliže se do okna nedá psát a není vidět již zmíněný otazník a pomlčku, za kterými již nejsou žádné další znaky, neukončili jsme řádně naše předchozí vyhodnocování. Jestliže jej chceme dokončit, postupujeme dál v odpovídání na otázky, které nám systém klade a po dokončení vyhodnocování, kdy jako poslední informaci od aplikace uvidíme oznámení o smazání dynamicky vložených dat, můžeme přejít z ručního vyhodnocování dat na čtení ze souboru a naopak.

Po zadání dotazu *go_file* (mezi slovy go a file je podtržítko a žádné mezery) se vyhodnotí najednou všechna data a obdržené výsledky se vypíší do okna Prologu i do výstupního souboru, který bude uložen ve stejném adresáři jako soubor vstupní.

Cesty k vstupnímu a výstupnímu souboru se dají změnit. Otevřete textový soubor `fc_form.pl` a najděte dva řádky, z nichž první začíná `file_in` a druhý `file_out`. Jsou uvedeny odpovídajícím komentářem a nacházejí se téměř hned na začátku uvedeného souboru. Upravte cestu k požadovanému souboru(`file_in` vstupní a `file_out` výstupní) dle vlastního uvážení. Cesta musí být uzavřena do apostrofů tak, jak jí ve výchozím stavu po otevření souboru najdeme.

7.2.3 Zadávání dat pod OS Linux



Obrázek 13: Ukázka prostředí SWI Prologu

Pokud jsme SWI Prolog nainstalovali s příslušného repozitáře, spustíme terminál (většinou funguje klávesová zkratka *Ctrl+Alt+T*) a pomocí příkazu *cd* přejdeme do složky, kde máme uloženou aplikaci, tedy soubor `fc_form.pl`. Pak spustíme SWI

Prolog napsáním příkazu *swipl* a mělo by se nám objevit prostředí podobné tomu na obrázku 13, na kterém je konkrétně vidět běžící SWI Prolog v okně terminálu v Xubuntu.

Abychom načetli aplikaci, napíšeme *[fc_form].* bez přípony `*.pl`. Za pravou hranatou závorkou musí být tečka. Poté již postupujeme při vyhodnocování dat naprosto stejným způsobem, který je uveden výše při popisu zadávání dat ve Windows.

Při načítání dat ze vstupního souboru musí být tento soubor taktéž ve stejné složce jako aplikace. To znamená soubor `fc_form.pl`. Stejný postup se týká i změny cest k vstupnímu a výstupnímu souboru.

Pro případné ukončení programu zavřeme okno terminálu.

8 Zhodnocení

Zhodnocení této práce nemohu začít jinou větou, než že vzhlížím k lidem, kteří ovládají programování v Prologu na vysoké úrovni. Je to oblast, která mne zajímá a vím o ní čím dál víc, ale stále se cítím na začátku a možná jednou budu patřit k těm, kteří Prologu opravdu rozumí a kdyby pročítali můj současný kód, pokyvovali by laskavě hlavami a mysleli by si: ,,Také to takhle jde udělat, ale moho by to jít i lépe.“

Na začátku všeho bylo zadání bakalářské práce, které, jak jsem si v úplném počátku myslel, bude jednoduché a praktické. Praktické určitě bylo. Došel jsem ale do fáze, kdy jsem o jeho jeho jednoduchosti začal více než pochybovat. Byl jsem již však natolik vtažen do problému, že jsem se na každý další krok v cestě ke stanovenému cíli těšil. Oblast programování mne vždycky zajímala a také jsem začínal vidět možnosti propojení toho, co dělám s podpůrnou rolí techniky, která by mohla v některých případech přerůst v téměř samostatnou činnost. Jelikož již teď vím, že základní funkce jsou hotové a použitelné, mohu říci, že co nejrozsáhlejší zautomatizování je do budoucna vlastně mé očekávání od dalších vývojových stádií aplikace vytvořené v rámci této práce..

Od začátku jsem postupoval dle zásady ,,účel světí prostředky“ a tím pádem tato aplikace i v takovéto podobě určitě splní svůj účel. Jsem o tom přesvědčen. Věnoval jsem spoustu času konzultacím s lékaři ohledně jejich postupu při provádění jejich poradenské činnosti s potenciálními i stávajícími klienty společnosti, se kterou spolupracuji. Ať již to byli lidé zdraví, kteří vyhledávali informace vedoucí k získání produktů pro preventivní péči o jejich zdraví, nebo lidé trpící určitými neduhy, kdy dotyční lékaři doporučují podpůrnou léčbu. Zdaleka ne všechny informace, které jsem do této doby získal, se promítly do vytvořené aplikace ve stavu, v jakém je v době odevzdání této práce. Částečně to bylo i mým úmyslem, jelikož je třeba ještě mnoha dalších setkání s příslušnými odborníky z oblasti zdravotnictví a bázi znalostí v současném stádiu je třeba považovat za ukázkovou a použitelnou pouze pro úzký okruh osob, které ji budou testovat do doby, než bude moci být plně nasazena. Roz-

hodně tedy nemohu doporučit široké veřejnosti slepě se řídit doporučeními, která vzešla z využití tohoto systému. V prvotní fázi budou vždy výstupy programu hodnoceny lékařem nebo jiným odborníkem do doby, než se podaří bázi znalostí doplnit natolik, že bude moci aplikace fungovat samostatně.

Okamžik, kdy jsem začal věřit, že by se to mohlo celé podařit nastal ve chvíli, kdy jsem objevil knihu [1]. I když se jedná o titul, který není úplně nejnovější a zvlášť v oblasti výpočetní techniky to většinou znamená, že knihy vydané byť jen před několika lety jsou již jen kronikami toho, jak to kdysi bývalo, našel jsem zde cenné informace a v podstatě i návod, jak začít s budováním znalostního systému postaveném na bázi Prologu.

Možná může mít někdo pocit, že se touto aplikací vracíme zpátky do sedmdesátých let, kdy se s počítači komunikovalo převážně textově. Nicméně i v této podobě nám ušetří spoustu času a umožní vybrat pro klienta produkty dle zkušeností lékařů, kteří je běžně doporučují na základě získání vstupních dat a následné konzultace.

V této chvíli je vytvářená aplikace ve stavu, kdy zajišťuje základní funkčnost, kterou jsem od ní již v samém zárodku očekával. Prvním z kroků k jejímu dalšímu vylepšování bude otestovat její kvality přesně tam, kde je očekáváno její nasazení. Mám výhodu v tom, že mohu sám být jedním z lidí, kteří mohou toto testování v přirozeném prostředí používání tohoto systému provádět. Na základě svých zkušeností věřím, že v budoucnu bude spousta zákazníků, kteří uvítají radu, jak si vybrat ten správný produkt a neutratit tak peníze zbytečně.

Jak jsem již předeslal, báze znalostí obsahuje zatím jen zlomek z počtu preparátů, které jsou v sortimentu dostupné. Byla vytvořena s úmyslem ukázat, jakým způsobem je zamýšleno tento systém používat a jak budou získávány informace od respondentů a jak s nimi bude v rámci systému nakládáno za účelem získání zamýšleného výsledku v podobě doporučení, které v budoucnu nemusí mít pouze formu konkrétních produktů, ale mohou se rozšířit i na doporučení změny stravovacích a pohybových návyků. Jelikož jsem se snažil systém co nejvíce zobecnit, domnívám se, že již nebude vyžadovat tak velké úsilí doplnit tuto bázi fakty a pravidly, která již v této chvíli mám k dispozici od lékařů, se kterými jsem se o mé práci od zadání až po odevzdání radil.

Již teď je více než jisté, že se mi spoustu věcí nepovedlo zvládnout v termínu, ve kterém bych si je zvládnuté představoval. Jednou z těchto nepříjemností je nedostatečně přívětivé prostředí pro komunikaci jak přímo s klientem přes web, tak i se správci dat a dalšími osobami s tímto programem pracujícími. Hodlám proto v tomto směru podniknout kroky nejen sám, ale již jsem domluven s několika kolegy, kteří jsou v současnosti zkušenými programátory, že mi budou nápomocni ve vytváření lepšího uživatelského rozhraní. Ideálním stavem by byla kombinace webového rozhraní a textové komunikace se systémem v přirozeném jazyce nebo alespoň v jazyce, který se přirozenému bude blížit. Když se podívám o hodně dál, dokážu si představit i komunikaci hlasovou. Ale to už je jiný příběh...

9 Závěr

V rámci této kvalifikační práce jsem vykročil směrem, který jsem si při definici zadání vytyčil. Základní aspekty tohoto zadání byly naplněny a v průběhu jejich naplňování se objevilo několik dalších cílů, které by bez započaté práce vůbec nespatřily světlo světa. Domnívám se ale, že počet možností, které v současné době pro budoucí použití a vylepšení vidím, stále není konečný. A jsem tomu rád, protože mít v budoucnu fungující program, sloužící k užitku člověka, který se chce dozvědět alespoň v orientační rovině relevantní informaci ohledně péče o své zdraví v oblasti doplnění běžné stravy, považuji za vhodnou nadstavbu v kvalitě služeb, které v současné době jsou klientům zmíněné společnosti poskytovány.

Reference

[1] MERRITT, Dennis. *Building Expert Systems in Prolog*. 1989, Springer-Verlag.

[2] MAŘÍK, Vladimír – ŠTĚPÁNKOVÁ, Olga – LAŽANSKÝ, Jiří. *Umělá inteligence*. 1993, Academia. ISBN 80-200-0496-3

[3] KASTNEROVÁ, Markéta. *Poradce zdravého životního stylu*. 2012, Nová forma. ISBN 978-80-7453-250-4

[4] SVAČINA, Štěpán a kolektiv. *Klinická dietologie*. 2008, Grada. ISBN 978-80-247-2256-6

[5] BLACKBURN, Patrick – BOS, Johan – STRIEGNITZ, Kristina. *Learn Prolog Now!* 2006 - 2012, Online.
`http://www.learnprolognow.org`

[6] DVOŘÁK, Jiří. *Expertní systémy*. Skriptum. 2004, VUT Brno.

[7] KRYL, Rudolf. *Úvod do programovacího jazyka PROLOG*. Verze 3.03, KSVI MFF UK Praha.

Příloha

Kód k příkladu v sekci 4.3

Příloha - kód k příkladu v sekci 4.3

```
go :-
    write('What is the patient\'s name? '),
    readln(Patient),
    hypothesis(Patient,Disease),
    write(Patient),
    write('probably has '),
    write(Disease),
    write('.'),nl.

go :-
    write('Sorry, I don\'t seem to be able to'),nl,
    write('diagnose the disease.'),nl.

symptom(Patient,fever) :-
    write('Does '),
    write(Patient),
    write(' have a fever (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,rash) :-
    write('Does '),
    write(Patient),
    write(' have a rash (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,headache) :-
    write('Does '),
    write(Patient),
    write(' have a headache (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,runny_nose) :-
    write('Does '),
    write(Patient),
    write(' have a runny_nose (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,conjunctivitis) :-
    write('Does '),
    write(Patient),
    write(' have a conjunctivitis (y/n) ?'),
    response(Reply),
    Reply='y'.
```

```
symptom(Patient,cough) :-
    write('Does '),
    write(Patient),
    write(' have a cough (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,body_ache) :-
    write('Does '),
    write(Patient),
    write(' have a body_ache (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,chills) :-
    write('Does '),
    write(Patient),
    write(' have a chills (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,sore_throat) :-
    write('Does '),
    write(Patient),
    write(' have a sore_throat (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,sneezing) :-
    write('Does '),
    write(Patient),
    write(' have a sneezing (y/n) ?'),
    response(Reply),
    Reply='y'.

symptom(Patient,swollen_glands) :-
    write('Does '),
    write(Patient),
    write(' have a swollen_glands (y/n) ?'),
    response(Reply),
    Reply='y'.


hypothesis(Patient,measles) :-
    symptom(Patient,fever),
    symptom(Patient,cough),
    symptom(Patient,conjunctivitis),
    symptom(Patient,runny_nose),
```

```
        symptom(Patient,rash).

hypothesis(Patient,german_measles) :-
        symptom(Patient,fever),
        symptom(Patient,headache),
        symptom(Patient,runny_nose),
        symptom(Patient,rash).

hypothesis(Patient,flu) :-
        symptom(Patient,fever),
        symptom(Patient,headache),
        symptom(Patient,body_ache),
        symptom(Patient,conjunctivitis),
        symptom(Patient,chills),
        symptom(Patient,sore_throat),
        symptom(Patient,runny_nose),
        symptom(Patient,cough).

hypothesis(Patient,common_cold) :-
        symptom(Patient,headache),
        symptom(Patient,sneezing),
        symptom(Patient,sore_throat),
        symptom(Patient,runny_nose),
        symptom(Patient,chills).

hypothesis(Patient,mumps) :-
        symptom(Patient,fever),
        symptom(Patient,swollen_glands).

hypothesis(Patient,chicken_pox) :-
        symptom(Patient,fever),
        symptom(Patient,chills),
        symptom(Patient,body_ache),
        symptom(Patient,rash).

response(Reply) :-
        read(Reply),
        write(Reply),nl.
```