Erdélyi Tudományos Diákköri Konferencia

Kolozsvár, 2008. május 23-24.

Szakértői rendszerek

Szerző: Régeni Ágnes

Babes-Bolyai Tudományegyetem

Matematika-Informatika kar

Informatika szak, III. évfolyam

Témavezető: Dr. Soós Anna, egyetemi docens

Babes-Bolyai Tudományegyetem

Matematika-Informatika kar

Numerikus Analízis és Statisztika Tanszék

Tartalomjegyzék

1. Bevezetés....3
2. A mesterséges intelligencia
2.1 A Darthmouth-i konferencia....4
2.2 Intelligens viselkedés....4
2.3 MI kutatások....5
3. Szakértői rendszerek
3.1. Általánosan a szakértői rendszerekről....6
3.2. Tudásalapú rendszerek....6
3.2.1 Tudásalapú rendszerek – szakértői rendszerek....6
3.2.2 Döntéstámogató és döntést hozó rendszerek....8
3.3 Szakértői rendszerek. Mikor?....8
3.4 Történeti áttekintés
3.4.1 DENDRAL....9
3.4.2 MYCIN....9
3.4.3 EMYCIN....10
3.5 Szakértői rendszerek felépítése....10
3.6 Tudásreprezentáció és következtetés....11
4. A JESS fejlesztői keretrendszer
4.1 JESS – Java Expert System Shell....12
4.2 Rete algoritmus....13
5. MeExSys szakértői rendszer
5.1 Az ötlet....16
5.2 Felhasznált technológiák....17
5.3 Eredmények....17
6. Összegzés....23
Irodalomjegyzék....24

1. Bevezetés

„Egy szakértői rendszert a számítógépben tárolt ismeretbázis megjelenítéseként értelmezünk, ahol a szakértői ismeretek olyan formában tárolódnak, hogy a rendszer képes intelligens tanácsot vagy intelligens döntést hozni az adott kérdéssel összefüggésben. További jellemző, amit sokan alapvetőnek is tartanak, a rendszernek az a képessége, hogy - igény szerint és az érdeklődő számára alkalmas módon - igazolja is saját okfejtését.” (British Computer Society's Specialist Group, 1983)

Napjainkban egyre több probléma megoldására használják az informatikát és az általa nyújtott lehetőségeket. A mesterséges intelligencia fejlődésével ez a kör egyre csak bővül. Dolgozatom célja, hogy rövid bepillantást nyújtsak ebbe a körbe és feltárjam a mesterséges intelligencia egyik, még fejlődésben levő ágát.

Ismertetem a szakértői rendszerek alaptulajdonságait, felépítésüket, történetüket, majd bemutatom az általam megvalósított, szívbetegségek diagnosztizálására használható szakértői rendszer felépítését, megvalósításának technikáit, működését.

Célom, hogy mindenki számára egy érthető meghatározást adjak a szakértői rendszerekről, azok előnyeiről és hátrányairól, alkalmazásairól.

2. A mesterséges intelligencia

2.1 A Darthmouth-i konferencia

A korai ötvenes évekre tehető a mesterséges intelligencia kutatásának a kezdete. Ekkor rendezett a Dartmouth College-ban John McCarthy egy konferenciát, amelynek témája az emberi gondolkodás folyamatát modellezni képes gép megalkotása volt. A konferencián jelen levő nagy tudósok voltak azok, akik a mesterséges intelligncia alapjait lefektették. Ezen tudósok közé tartozott Allen Newell, Herbert Simon, Marvin Minsky, Nathaniel Rochester, Claude Shannon. A tíz tudósból álló csoport célkitűzése az volt, hogy két hónap alatt választ adjon azon feltételezésre, mely szerint bármilyen intelligens viselkedést lehet annyira pontosan definiálni, hogy a leírások alapján egy gép szimulálni tudja azt. A gépi tanulás, beszélés, problémamegoldás és egyéb olyan problémák elméleti alapjait, amelyek megoldása az emberek sajátja volt. Az azóta eltelt időben a mestersége intelligencia kutatása önálló tudománnyá vált, amelynek számos területe alakult ki és több, jól elkülöníthető irányzat jött létre.

2.2 Intelligens viselkedés

A mesterséges intelligencia célja olyan eszközök készítése volt, amelyek a külvilág fele intelligens viselkedést mutatnak. Tehát a cél az intelligens problémamegoldás. De mitől lesz egy viselkedés intelligens? Az értelmi szint, működés fokmérőjeként emlegetik az intelligenciát, amely összefügg az előzőleg szerzett tapasztalati anyag alkalmazásával és a gondolkodóképességgel.

Az intelligencia nehezen definiálható, mert mindenki mást tekint intelligens viselkedésnek. Emiatt van az is, hogy a mesterséges intelligenciát is szinte lehetetlen pontos módon definiálni. Ennek ellenére léteznek lényeget megragadó, elfogadható megfogalmazások az intelligenciát és a mesterséges intelligenciát illetően:

„*Az intelligencia annak a képességnek a birtoklása, mellyel szükség esetén képesek vagyunk alkalmazkodni a környezetünkhöz vagy úgy, hogy saját magunkat, vagy a környezetet változtatjuk meg, illetve, ha ez nem lehetséges, akkor egy új környezetet keresünk.*” (Encyclopedia Britannica, 2001)

„*Az a tudomány, melynek segítségével olyan cselekedetek végrehajtására teszünk képessé*

gépeket, melyek véghezvitele emberi intelligenciát igényelne." (Minsky gondolata a mesterséges intelligenciáról)

Levonhatjuk a következtetést, hogy a mesterséges intelligencia annak a kutatása, hogy a számítógépek hogyan oldhatják meg hatékonyabban az olyan problémákat, amelyeket jelenleg az emberek oldanak meg a leghatékonyabban, a legjobban. Viszont mikor viselkedik intelligensen egy rendszer? Amikor egy adott cél elérése érdekében a rendelkezésre álló ismeretek, viselkedések alapján a legjobb döntést hozza, miközben csak a szükséges és elégséges tevékenységeket végzi el.

2.3 MI kutatások

A mesterséges intelligencia egy olyan kutatási terület, amely arra irányul, hogy a számítógépek olyan tevékenységeket tudjanak végezni, amelyek intelligensnek tekinthetőek. A kutatásoknak két fő irányát különíthetjük el: kognitív és számítástudományi.

A *kognitív* kutatások a számítógépet eszközként használják, az emberi gondolkodást, problémamegoldást modellezik annak segítségével. Itt is elkülöníthető több modellezési módszer. Megemlíteném a „felülről-lefele" megközelítést, amely viselkedési adatokból indul ki és megpróbál egy olyan algoritmust vagy modellt találni, amely visszaadja a kiinduló jelenségeket. Létezik az „alulról-felfele" modellezési stratégia is, amely ezzel ellentétben előbb modelleket alkot az anatómiai ismeretek alapján, és így próbálja előállítani a megfigyelhető jelenségeket a számítógépek segítségével.

Ezzel szemben a *számítástudományi* megközelítés a számítógépet nem mint eszköz használja fel, hanem mint a kutatás céljának alanya, hiszen „viselkedésének" intelligensebbé tételét próbálja megvalósítani.

Dolgozatom további részében a mesterséges intelligencia kutatások egyik alkalmazásával, a szakértői rendszerekkel foglalkozok.

3. Szakértői rendszerek

3.1 Általánosan a szakértői rendszerekről

A bevezetőben levő idézet óta a szakértői rendszereknek pontosabb meghatározásai születtek. Szűkebb és tágabb értelemben egyaránt definiálhatjuk a szakértői rendszereket. Szűkebb értelemben egy olyan eszköz, amely könnyebbé és hatékonyabbá teszi a programfejlesztést. Tágabb értelemben az első lépést jelenti abba az irányba, hogy a programozás a numerikus, direkt programozásnál magasabb szintre kerüljön.

A szakértői rendszer egy, különböző mesterséges intelligencia módszerekre épülő problémamegoldó rendszer, amely tartalmazza egy szűkebb, jól definiált problématerület ismereteit. Nagy méretű, komplex problémák megoldásában nyújthat segítséget, figyelembe véve az emberi szakértők problémamegoldási folyamatát. A „valódi", emberi szakértőkhöz hasonlóan az eredményhez való eljutás érdekében alkalmazza a heurisztikus módszereket.

Összehasonlítva a humán szakértőt és a szakértői rendszert, megállapítható, hogy míg az emberi szakértő képessége és tudása idővel változik, felkészítése hosszú és költséges folyamat, a döntéshozatalát számos, külső tényező befolyásolhatja, addig a szakértői rendszer tudása állandó, fejlesztése drága, de sokszorosítása, használata és karbantartása olcsó és viszonylag könnyű, állandó és megismételhető eredményt biztosít.

Az intelligens rendszereknek azonban hátrányaik is vannak: csak szűk területen, speciális problémák megoldására használhatók, nem tud józan ésszel „gondolkodni" a tudat hiánya miatt, nem veszi észre saját határait.

A megnevezés, hogy „szakértői rendszer" tehát félrevezető, ugyanis nem helyettesítik az emberi szakértőt, csupán arra vállalkoznak, hogy a sok specifikus ismeretet és a nagy figyelmet meg tudást igénylő munkájából magukra vállalják a mechanizálható részeket.

3.2 Tudásalapú rendszerek

Tömören megfogalmazva, a tudásalapú rendszerek feladata egy adott területen előforduló probléma megoldására egy javaslatadás vagy a probléma okának kiderítése. Azt is meg kell tudnia mondani, hogy hogyan következtetett és a megadott információkból hogyan jutott el az adott eredményhez - abban az esetben, ha magyarázó alrendszert is építettek bele. Ez nem

szükséges ahhoz, hogy a kifejlesztett rendszert hatékonyan alkalmazhassuk, viszont a tesztelésnél hatékony lehet a hibák kiderítésére, és a felhasználónak is bizonyíthatja következetességét. A tudásalapú rendszerek szimbolikus adatokkal dolgoznak a numerikus adatok helyett.

Mit is nevezünk tudásalapú rendszereknek? Egy olyan alkalmazást, amelyben az ismeretek, a szaktudás a rendszer működése során az általános problémamegoldó algoritmusoktól jól elkülöníthető helyen, a tudásbázisban tárolódnak. A többi MI programok közül jól elkülöníthetőek, hiszen a programstruktúrájuk más. A következtetést megvalósító algoritmusok általánosak, függetlenek az adatoktól, amellyel majd dolgoznak.

3.2.1 Tudásalapú rendszerek és szakértői rendszerek

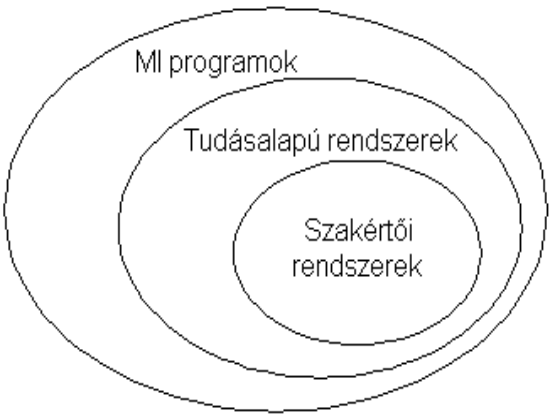
A szakértői rendszerek a tudásalapú rendszerek részét alkotják. A szakértői rendszernek egyesek szűkebb jelentést tulajdonítanak a tudásalapú rendszerek jelentésénél. Mások pedig szinonimaként használják a két megnevezést. Azon források, amelyek elkülönítik a két fogalmat, úgy fogalmaznak, hogy abban az esetben beszélhetünk szakértői rendszerről, amikor a tudásbázisban tárolt ismeretek egy jól meghatározható, speciális terület szakismereteit írják le, tudásalapú rendszerek esetében a tárolt ismeret, tudás, információ általános. Például, ha egy orvos ismereteit vesszük és a rendszerünket az ő ismerete, tudása köré építjük, akkor szakértői rendszerről beszélhetünk, hiszen az ismeretek egy adott szakterülethez, az orvoslási szakterülethez kötődnek. Ha viszont a közlekedéshez szükséges ismeretek köré építjük a rendszert, akkor tudásalapú rendszerről beszélhetünk, mivel ezen ismeretek általánosak, szinte mindenki ismeri, nem feltétlenül kell szakértőnek lennie a közlekedésben.

Feltevődik viszont a kérdés, hogy melyik a helyes megközelítés, mi is a kapcsolat a tudásalapú és a szakértői rendszer között. Mindkét megközelítés helyes, hiszen ha a szerkezeti modellt figyeljük, mind a tudásalapú és szakértői rendszerek estében szinte teljesen megegyeznek. Tehát, ha a technológiai megoldásokról, implementációról beszélünk, a két rendszert egyként kezelhetjük. A következtetési, magyarázó és más, kizárólag technikai jellegű algoritmusok felépítése ugyanaz a két rendszerben, mivel ezek teljesen függetlenek a tárolt ismeretektől.

Ezzel szemben ha a rendszerek alkalmazási területéről vagy az ismeretek jellegéről beszélünk, jobb különválasztani a szakértői rendszert a tudásalapú rendszertől.

Tehát a mesterséges intelligencia programok intelligens problémamegoldó eszközök. Ilyen problémák pl. a tanulás, természetes nyelvű közlés, képértelmezés stb. A tudásalapú rendszerek

ezen programok egy részét jelentik, amelyeknél elkülöníthető a tudásbázis. A szakértői rendszer egy tudásalapú rendszer, ahol a tudást az emberi szakértő biztosítja. A rendszer az ember által birtokolt tudás egy nagy részét kell, hogy tartalmazza ahhoz, hogy elfogadható, a szakértő által is adott eredményt adjon.



3.2.2 Döntéstámogató és döntéshozó rendszerek

A szakértői rendszereknek alapvetően két típusuk különböztethető meg: döntéstámogató és döntéshozó rendszerek. Ez a két csoport annak alapján különíthető el, hogy mennyire, milyen mértékben támaszkodhatnak az általuk adott tanácsokra az adott alkalmazási területen, ahol használják. Ez szerint a döntéstámogató rendszerek felhasználóit aktív felhasználóknak, míg a döntéshozó rendszerek felhasználóit passzív felhasználóknak tekinthetjük.

A döntéstámogató rendszerek emlékeztetik az emberi szakértőt a megfontolandó következményekre, alternatívákra, amelyekre ő esetleg nem gondolt és elsiklana felettük a döntéshozatal során. Az orvostudományban használt szakértői rendszerek, amelyek diagnosztizáló jellegűek, adnak tipikus példát az első típusra. Ezen területen feltehetőleg nem lesznek olyan rendszerek, amelyek döntéshozóként fognak alkalmazni.

A döntéshozó szakértői rendszerek segítséget nyújtanak a problématerületen kevésbé jártas felhasználónak a probléma megoldásának megtalálásában. Nem tapasztalt embereknek olyan területeken is engedélyezik a döntéshozást, ahol tapasztalatlanok, szakképzetlenek.

3.3 Szakértői rendszerek. Mikor?

Mikor támogatható valamilyen problématerület szakértői rendszer segítségével? Ha a problématerület *szűk*, de viszont *bonyolult*, és meg lehet benne ragadni olyan tudást, amely nem mindenki számára kézenfekvő. A problématerületnek rendelkeznie kell emberi szakértőkkel

akiknek a tudásából ki lehet indulni a rendszer elkészítéséhez, és ezen szakértők között a szakterület alapkérdéseiben nagyfokú egyetértés kell legyen. Számos tanpélda, alapadat kell létezzen az adott szakterületen, ahhoz, hogy a rendszer tudásának korlátait meg lehessen határozni, illetve tesztelni lehessen. Ezeken kívül, annál jobb szakértői rendszert lehet építeni egy adott szakterületen, minél jobban feloszthtató a terület részproblémákra, amelyek kis mértékben zavarják egymást, csak nagyon kicsit interferálnak.

3.4 Történeti áttekintés

3.4.1 DENDRAL (1965-1983)

A DENDRAL volt a legelső szakértői rendszer. Fejlesztői azt akarták megvalósítani, hogy rendszerük alkalmazni tudja a tudományos ismereteket és a következtetéseket a szerves kémia tudomány területéről. Céljuk volt egy olyan technológia létrehozása, amellyel képesek jobban megérteni a szakterület alapvető problémáit. Ezt a rendszer által szolgáltatott megoldások magyarázásának a lehetősége biztosítja. Képes volt az ismeretbázis alapján eddig ismeretlen szerves anyagok felépítésének meghatározására, képessége sok esetben az emberi szakértőével vetekedett. Az elkészítéshez vegyészek, biológusok és informatikusok egyaránt hozzájárultak és megszületett az első szabály-alapú szakértői rendszer, a DENDRAL, amelyet az egyetemeken és az iparban egyaránt használtak.

5.4.2 MYCIN (1972-1980)

A MYCIN a vér bakteriális fertőzéseinek diagnosztizálására készült. Megállapította a betegséget bizonyos valószínűséggel és javaslatot tett a gyógykezelésre is. Ezen kívül részletes magyarázattal szolgált arról, hogy hogyan jutott el az adott következtetésre. Eredményei elérték az orvos szakértők eredményeit, szabályozott körülmények között.

Itt található meg elsőként a tudásbázis és a következtető mechanizmusok szétválasztása, szabályalapú, rendszere a hátraláncolásos következtetést használta. Tudásbázisa könnyen bővíthető újabb szabályokkal. A MYCIN szakértői rendszerbe bizonytalanságot jelző faktorokat is beépítettek, mellyel jelzi az adott következtetés megbízhatóságát. Mintájára született a THEIRESIAS, CENTAUR, GUIDON stb.

5.4.3 EMYCIN

„Essential MYCIN” elnevezésből jön az EMYCIN név, a rendszer maga a MYCIN következtető motorját használja, viszont a tudásbázis nélkül. A MYCIN tudás nélküli váza. Egy tudásbeszerzést megkönnyítő alrendszerrel bővítették ki. A felhasználónak a MYCIN legfontosabb részeit kínálja fel, amelyek szakterülettől függetlenek, ezzel egy keretrendszert (shell-t) alkotva. Segítségével újabb szabály-alapú szakértői rendszerek készíthetők. Ezek probléma típusai azonban a diagnosztizálás problémájához kell hogy hasonlítsanak.

3.5 Szakértői rendszerek felépítése

Egy szakértői rendszer elkészítésében két kulcsfigurát kell feltétlenül megemlíteni. Az egyik a *tárgyköri szakértő*, a másik a *tudásmérnök*.

A tárgyköri szakértő általában nem rendelkezik informatikai ismeretekkel, viszont az ő ismereteit, szakértelmét próbáljuk az alkalmazásba átvinni. Ezt a folyamatot nevezzük tudásbeszerzésnek. Feladata ezek után sem szűnik meg, hiszen a rendszer elkészítése után a érvényesítése következik, amit szintén a tárgykörű szakértőnek kell elvégeznie.

A tudásmérnök az informatikai tudását felhasználva formalizálja a tárgykörű szakértő ismereteit, rendszerezi, majd elkészíti a rendszert a kiválasztott fejlesztő eszközök segítségével.

Nagyobb méretű szakértői rendszerek létrehozásánál a tárgyköri szakértők és tudásmérnökök száma is növekedhet.

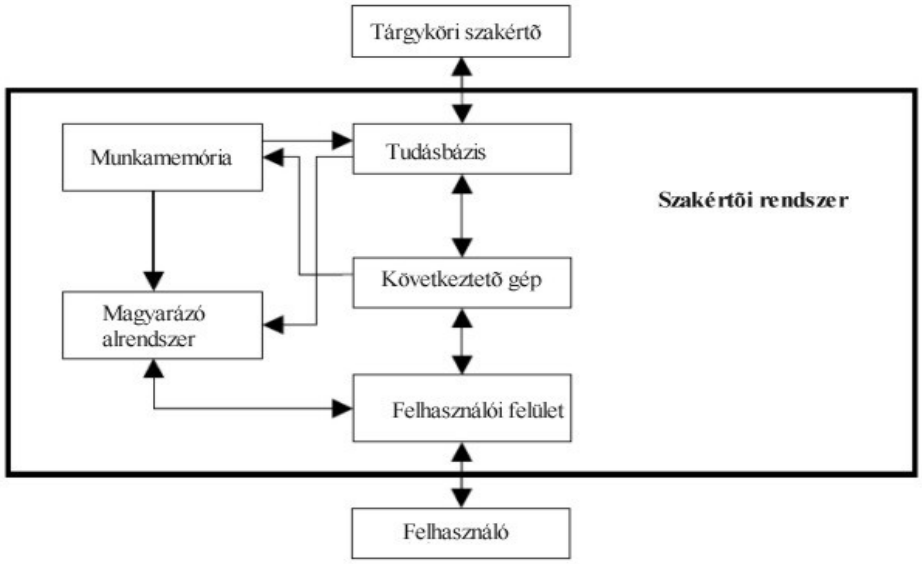
A szakértői rendszerek (tudásalapú rendszerek) legfontosabb részei: tudásbázis, munkamemória, következtető gép. Ezek mellet még implementálható egy magyarázó alrendszer is, amely sokszor elősegítheti a fejlesztés folyamatát.

A rendszer tudását a tudásbázis tartalmazza. Ezt a tudásmérnök állítja össze és tartja karban, felhasználva továbbra is a tárgyköri szakértő ismereteit.

A kommunikáció a felhasználó és a szakértői rendszer között a felhasználói felületen keresztül történik. Itt tudja leírni, meghatározni a problémát és itt kapja meg a választ, a megoldást rá.

A felhasználó által leírt probléma a munkamemóriába kerül. A megoldás során a következtető gép, felhasználva a tudásbázisban tárolt adatokat, megpróbál választ adni. A következtetést a magyarázó alrendszer segítségével tudjuk nyomon követni.

Léteznek úgynevezett szakértői keretrendszerek (shell-ek), amelyek a szakértői rendszer vázát tartalmazzák. A fejlesztés idő- és munkaszükséglete így jóval lerövidül.



3.7 Tudásreprezentáció és következtetés

A következtetési mechanizmus és a tudásreprezentáció aligha elválaszthatóak egymástól. Ha úgy döntünk, hogy egy bizonyos tudásreprezentációt használunk, akkor ezzel már meghatároztuk a következtetésre használható algoritmusok körét is.

Akármilyen reprezentációs nyelvet is választunk, ezt hatékonyan fel kell hogy tudjuk használni: ezért legyen kifejező és tömör, miközben független az általa reprezentálni kívánt tudástól. Az általa leírt információ legyen felhasználható a következtetésre, mindemellett legyen egyértelmű.

A legfontosabb reprezentációs formák:

- keretek
- szabályok
- esetek
- heurisztikák

Legáltalánosabban a szabály-alapú tudásreprezentációt alkalmazzák, mivel az áll legközelebb az emberi gondolkodáshoz. Sokszor döntünk mi is a ha – akkor szabályokat használva.

4. A JESS fejlesztői keretrendszer

4.1 JESS – Java Expert System Shell

A JESS elnevezés a Java Expert System Shell rövidítése, azaz egy szakértői rendszer fejlesztői keretrendszert jelent Java platformra.

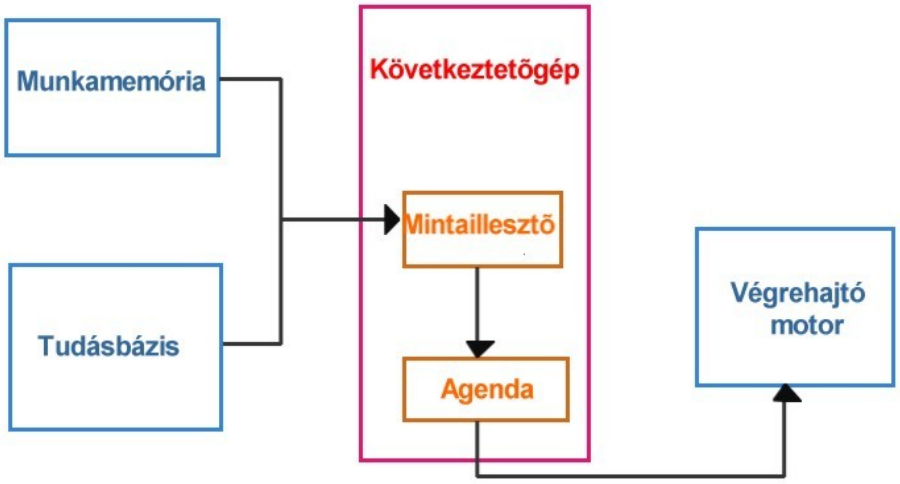
A JESS végül is egy következtető motor és a kódolás környezete, amelyben a tudásbázist megírjuk. Teljes egészében a Java programozási nyelvben íródott, fejlesztése a késő 90-es években kezdődött a „Sandia National Laboratories”-ban (Livermore, Kalifornia) Dr. Ernest Friedmann-Hill által. Alapjául a CLIPS programozási nyelv állt, amelyet a NASA által fejlesztettek ki az 1980-as években, szintén szakértői rendszerek létrehozása céljából. A JESS fejlesztési folyamata azonban nem fejeződött be, hiszen mindig a legutolsó, stabil verziót fejlesztik egy próba verzióvá, amelyből majd az új stabil verzió jön létre az esetleges hibák kiszűrése után. Ezek megtalálásában természetesen a felhasználók, a szakértői rendszert fejlesztők is nagyban hozzájárulnak.

Ingyenesen letölthető a http://jessrules.com/ oldalról a 30 napos verzió, ezenkívül licenc is kérhető (ingyenes vagy fizetett attól függően, hogy kereskedelmi vagy tudományos használatra kérjük).

A JESS segítségével olyan Java szoftvereket írhatunk, amelyek képesek következtetni, felhasználva az általunk megadott tudást. A tudásbázist szabály-alapú reprezentációval adhatjuk meg. Tartalmaz egy teljesen felépített fejlesztői környezetet az Eclipse platformra. A JESS a Rete algoritmust alkalmazza a szabályok levezetésére. A JESS megváltoztathatja illetve következtethet a Java objektumokról, ezenkívül egy erős kódolási környezetet biztosít, amelyben például Java osztály-objektumokat hozhatunk létre stb. Végül is, Javában programozhatunk anélkül, hogy Java kódot írnánk.

Mint tudjuk, egy szakértői rendszernek rendelkezni kell egy következtető motorral is a tudásbázis és a munkamemória mellett. A JESS esetében is itt történik a *mintaillesztés* (a Rete algoritmus alapján kiválasztja a megfelelő szabályokat), a *szabály-kiválasztó mechanizmus* („agenda”, amely a kiválasztott szabályok közül meghatározza a legerősebbet), és a *végrehajtó motor*, ami a szabályok aktiválásáért felelős. A következtető folyamat először illeszt, majd kiválaszt és végül végrehajt.

Az alábbi ábra a JESS architektúra diagrammját mutatja.



4.2 Rete algoritmus

A Rete algoritmus egy hatékony minta-felismerő algoritmus, amelyet Dr. Charles L. Forgy tervezett, és először 1974-ben publikált és később, 1979-ben a doktori tézisében adta ki. A JESS kereten kívül még a CLIPS, a SOAR is ezt az algoritmust használják.

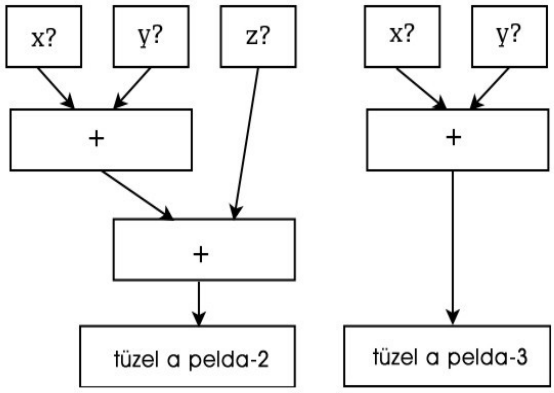
De hogyan is alkalmazza a Rete algoritmust a JESS? Ezt egy példa bemutatásával szemléltetem:

```
(defrule pelda-2
    (x)
    (y)
    (z)
    => )
```

```
(defrule pelda-3
      (x)
      (y)
      => )
```

A Rete algoritmus épít egy csomópont-hálózatot, amelyek mindegyike egy vagy több tesztet tartalmaz a szabály bal oldaláról. A hálózat alján a csomópontok különálló szabályokat reprezentálnak. Amikor valamilyen tények halmaza leért a hálózat aljára (tehát a teszteken sikeresen átment), akkor ezt a halmazt aktiválni kell, és a talált szabály jobb oldalát végrehajtani.

A fenti példát felírva a csomópontokra a következő ábrát kapjuk:

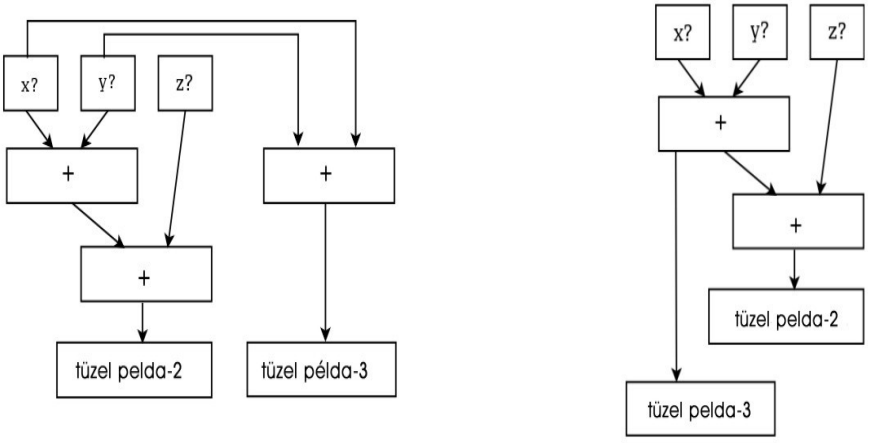


Az x?, y?, z? csomópontok tesztelik, ha egy tény tartalmazza az adott adatokat, a + csomópontok megjegyzik az összes tényt a szabály későbbi, lehetséges tüzelésére abban az esetben, ha a bal oldalról és jobb oldalról is kaptak adatokat.

A Rete egyik optimalizálása, hogy megosztja a közös csomópontokat a minta-hálózatban.

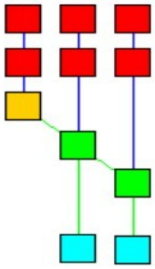
Észrevehető, hogy az egyik összekötő csomópont kétszer szerepel és ugyanaz a szerepe mindkét esetben, az x és y párokat vonja össze. Így a minta és az összekötő-hálózatok helyesek és csak fele méretük van az eredetihez képest.

Ezt a két lépést a lenti ábrák mutatják:



A JESS lehetőséget nyújt arra, hogy megtekintsük a Rete hálózatunkat.

Az alábbi hálózat a pelda-2 és pelda-3 szabályokba való belépéskor jön létre. A csomópontok sokszínűek, elkülönítve a típusukat.



Minden csomópont a jess.Node osztályból van származtatva, és ennek több alosztálya van: Node1, Node2, NodeNot2, NodeJoin, NodeTerm.

A Node1 egy utasítás taggal is rendelkezik, amely másképp viselkedik különböző tesztek esetében. Például Node1 típusú amelyik az első mezőt teszteli egy tényből, vagy a mezők számát stb. Ezek a piros színűek. A Node2 típusúak zöldek, ezek nem a Node1 típusúak, tehát nincsenek bennük utasítások. A NodeNot2 csomópontok színe sárga. Kékek a Defrule típusú csomópontok, azaz a szabályok.

A Rete algoritmus a sebességet helyezi előtérbe a hellyel szemben, de ez nem annyira jelentős nagyobb alkalmazások esetén sem.

A gyorsaság és a használhatóság nagyban függ attól, hogy hogyan tervezzük a szabályainkat.

5. MeExSys szakértői rendszer

5.1 Az ötlet

Az általam fejlesztett szakértői rendszernek a neve is tartalmazza annak angol meghatározását: Medical Expert System.

Külföldön való tartózkodásom során egy olyan kurzus hallgatója voltam, amely az orvosi informatikával foglalkozott. Lépésről lépésre vezetett be ebbe a tudományba és bepillantást nyújtott az informatika segítségével az orvostudományban elért eredményekhez. Ekkor ragadott meg a szakértői rendszer és az eddig, általa elért eredmények. Mivel az informatika mellett mindig is érdekelt az orvostudomány, természetesnek tűnt egy olyan téma, amely e kettőt összekapcsolja.

Mivel nem láttam még komolyabb szakértői rendszert, egy ilyen rendszer megírását tűztem ki célul. Szükségem volt azonban egy keretre, amely segítséget nyújt a következtetésben, és ki kellett válasszam a legmegfelelőbbet, amely a tudásreprezentációmhoz illik.

A betegségeket általában szabály-alapú reprezentációval írják le, hiszen ez így a leghatékonyabb. Az orvos is, amikor egy betege érkezik, sorba kérdi tőle a kérdéseket, amelyek a tüneteire vonatkoznak. Feltesz egy kérdést, a következő kérdést pedig az előbbire adott válasza alapján teszi fel. Nem kérdezhet össze-vissza és nem kérdezhet olyan kérdést sem, amely egyáltalán nem illik a képbe. A szabály-alapú reprezentációban lehet heurisztikákat is használni, hiszen mindig van egy olyan pont, amely elkülönít bizonyos betegségeket.

A szívbetegségek esetében ennek az alkalmazása szinte elkerülhetetlen, mivel a tünetek nagyjából megegyeznek. Tehát vannak közös tünetek. Viszont nem minden tünet közös. Például ha légszomjat érzünk (a rendszeremben a shortness of breath vagy dyspnea néven szerepel), akkor az már nem közös tünet, mivel ennek több formája van, és attól függően, hogy milyen típusúnak írja le a beteg a légszomját, különíthetőek el a különböző betegségek.

Ha a légszomj már krónikussá vált, akkor bizonyos betegség-családba irányítja a felhasználót a kérdések segítségével, ha viszont spontán keletkezik, akkor más családbeli betegségek valószínűsége nagyobb.

Végül az elvárt végeredmény egy szívbetegségeket bizonyos valószínűséggel diagnosztizálni tudó szakértői rendszer.

5.2 Felhasznált technológiák

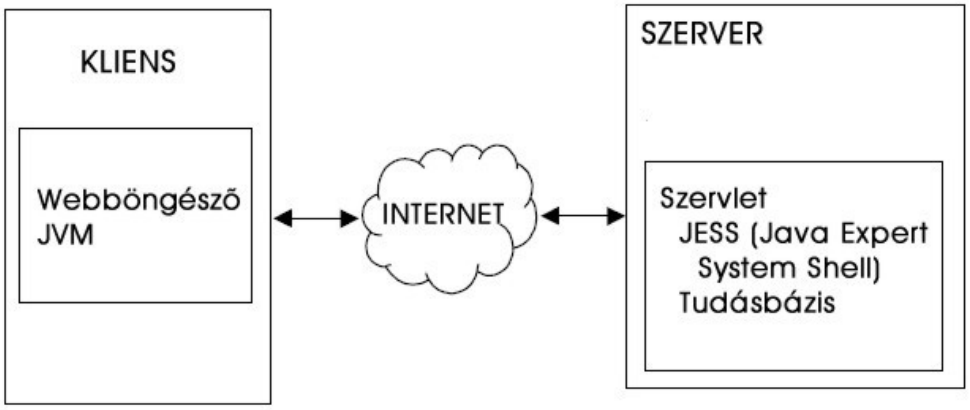
A fejlesztés elkezdésekor kétségek között voltam, hogy a CLIPS vagy a JESS fejlesztői keretrendszert válasszam. Végül azért döntöttem a JESS mellett, mert a CLIPS-hez képest fejlettebb és mivel még mindig fejlesztés alatt áll, mondhatni fejlődő rendszer, lehetséges a jövőben újabb funkcionalitásokkal is bővíteni az alkalmazást. Ezen kívül mindkettő a szabály-alapú tudásreprezentációt követi.

A Java programozási nyelvet és Eclipse platformot használok, természetesen a beépített, legutóbbi stabil verziójú JESS-sel (Jess70p2).

Mi is a szakértői rendszerek egyik lényege? Hogy az emberek elérhessék őket, ha a szakértő nincs a közelben. Innen is jött az ötlet hogy webes alkalmazást írjak.

Szükség van a kliens-szerver architektúrára is. A szerver oldalon van tárolva a tudásbázis és a JESS, míg a kliens csupán a böngészőjéből elérheti a szakértői rendszert.

A szerver a web-szerver, amely a klienssel való kommunikációt kezeli. A szervlet kezeli a kliens által adott adatokat, az továbbítja a keretnek.



5.3 Eredmények

Ahhoz, hogy a szakértői rendszer eredményesen működjön, négy Java osztályra volt egyelőre szükségem. Ezek mellett a .clp állományra, amely a tudásbázis, szabály-alapú reprezentációval megadva.

A Java osztályok: MeExSysServlet (a vezérlő szervlet osztálya), MeExSysReteControl (a

következtetési mechanizmus, azaz a Rete motor futását szabályozó osztály), MeExSysQuery (segítségével történik a html felület megjelenítése az újabb kérdéssel), MeExSysFinal (akkor hívódik meg, amikor eredményhez jutott a rendszer).

Ezen osztályokon kívül még .jsp állományok is vannak, amelyeknek a szakértői rendszer elindítása előtt van szerepük.

Hogy hogyan is működik az alkalmazás? A felhasználó elindítja a weboldalt, majd elindítja a MeExSys-t. Ekkor kezdődik el a kérdések feltevése.

Mivel a szívbetegségek tünetei általában megegyeznek (ezek a tünetek nem a fő tünetek, hanem a mellék, nem annyira lényeges tünetek, de mégis jelen kell lenniük annak érdekében, hogy a betegséget nagyobb valószínűséggel diagnosztizálni lehessen), így az indításkor a kérdések ezekre a tünetekre irányulnak. Akármilyen betegség irányába is vezetne a következtető motor, ezen tüneteket mindig megkérdezné, és így én az elején kérdem őket meg. A választól függetlenül jön a következő kérdés. Ezek a kérdések általánosak, amiket az orvos is megkérdez, ha egy beteg érkezik hozzá. Például, hogy érez-e fáradtságot mostanában, észrevett-e valamilyen változást a tömegével kapcsolatosan, érez-e émelygést és egyéb általános célú kérdések. Egymástól függetlenek, de lényegesek. Ezen kérdések után alkalmazok egy „ökölszabályt”, amely elkülönít néhány betegséget. Ez a kérdés a fent is emlegetett légszomjra irányul.

```
(defrule first-real-query-1
  (or
    (attr swelling-a-l-a)
    (attr swelling-a-f-a-b)
    (attr swelling-l-e)
    (attr swelling-not)
  )
=>
  (new MeExSysQuery "Have you felt a shortness of breath currently?
      (dyspnea)" 3
     (create$
        "yes" "(attr dyspnea-felt)"
        "not my main symptom" "(attr dyspnea-not-main)"
        "no" "(attr dyspnea-not-main)"
     )
   (fetch "ReteControl")
  )
)
```

Ez után történik a betegségek elkülönítése. Ennek a szabálynak a következtében egy MeExSysQuery típusú osztály-objektum jön létre. Miután ez létrejött, megtörténik a kiírás a html oldalra, és megjelenik három válaszlehetőség. Az egyik a „yes”, a másik a „not my main symptom” míg a harmadik a „no” lehetőségeket jelenti. Ha az elsőt választjuk, akkor a

munkamemóriába egy újabb tény kerül, amely jelzi, hogy a beteg érzett légszomjat és ez akár a fő tünete is lehet, mivel nem a másodikat választotta. Abban az esetben csupán melléktünetként lenne kezelve a légszomj. Tehát, választja az elsőt. Ekkor a Rete algoritmussal megkeresi az adott tényekre illeszkedő szabályt, tehát amikor ez a tünet megjelenik.

```
(defrule dyspnea-felt-query
  (attr dyspnea-felt)
=>
  (new MeExSysQuery "Does it appear suddenly or it became chronic?" 2
    (create$
      "it appears suddenly" "(attr dyspnea-suddenly)"
      "it became chronic" "(attr dyspnea-chronic)"
    )
    (fetch "ReteControl")
  )
)
```

Ennek a szabálynak, mint látjuk, a feltétel oldalán csupán az szerepel, hogy érezte a légszomjat, tehát, pont erre a mintára illeszkedik a tény. Ekkor folytatódik a lánc, és újabb kérdés tevődik fel. A kérdések irányítják a felhasználó újabb kérdéseit, míg eljuttatják egy betegséghez.

A tudásbázis nagy része a fenti szabályhoz hasonló szabályokból áll össze.

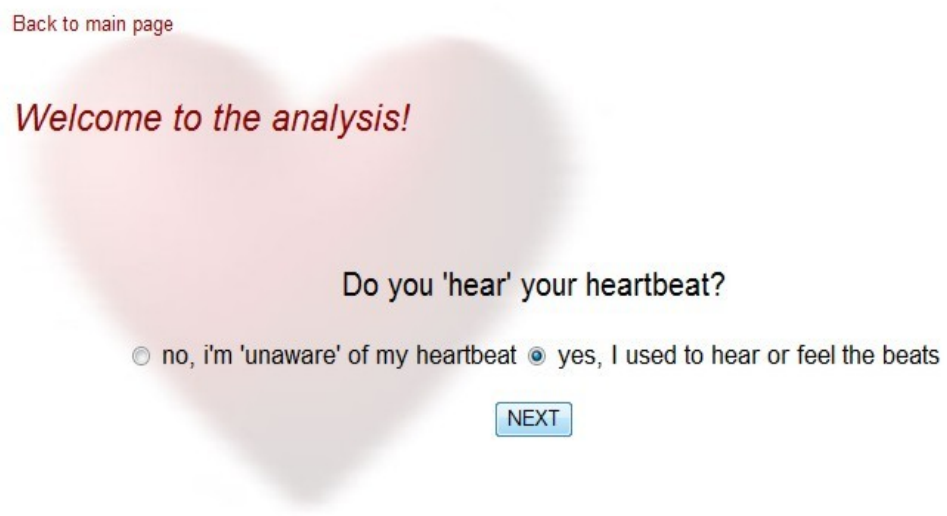
Most nézzük meg egy kissé bonyolultabb szabály Rete ábrázolását. Adott az alábbi szabály:

```
(defrule congenital-heart-disease-2
  (and
        (family congenital-heart-disease)
        (or
            (attr exercise-limited)
            (attr exercise-normal)
         )
   )
=>
  (new MeExSysQuery "Do you 'hear' your heartbeat?" 2
    (create$
      "no, i'm 'unaware' of my heartbeat" "(attr palpitations-not)"
      "yes, I used to hear or feel the beats" "(attr palpitations-felt)"
    )
    (fetch "ReteControl")
  )
)
```

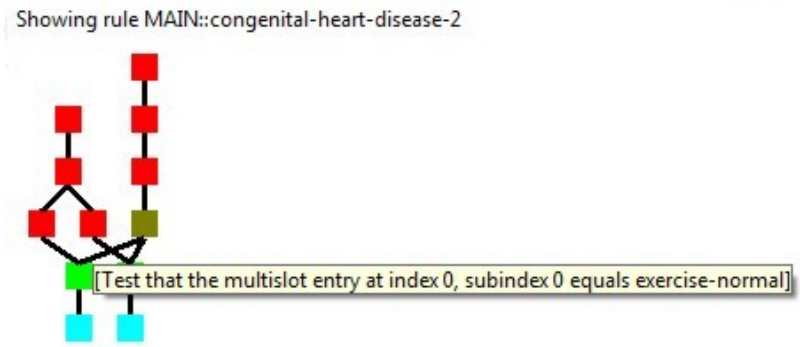
Ez a szabály akkor fog bekerülni a munkamemóriába, ha a család a „congenital-heart-disease” (veleszületett szívbetegség) és rendelkezik a két attribútum valamelyikével.

Amikor elérte ezt a szabályt, akkor már alkalmazta az „ökölszabályt” és annak eredményeképpen jutott el a veleszületett szívbetegségek családjába.

Ha ez a szabály alkalmazható, tehát a megadott tények alapján kiválasztható, akkor létrehoz egy új objektumot a MeExSysQuery osztályból és megjelenik a kérdés: „Do you 'hear' your heartbeat?”

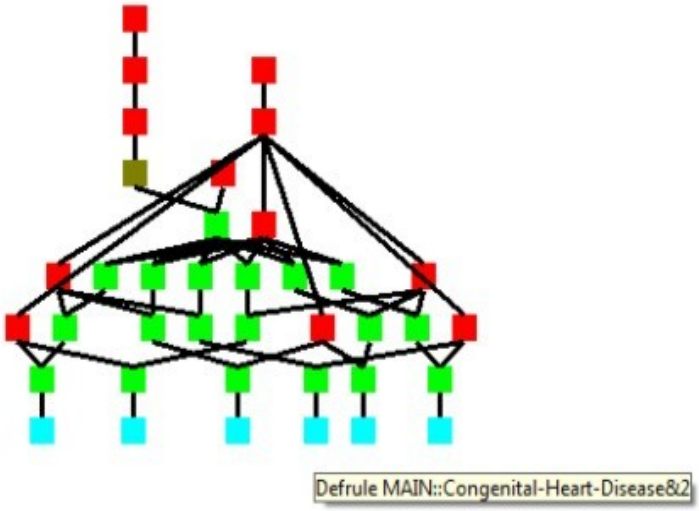


A fenti kép a szabály tüzelésének (végrehajtásának, aktiválásának) eredményét mutatja.
De mi is történik a megjelenítés mögött, a Rete-hálózatban? Erre a szabályra a Rete view (a JESS egy beépített View-ja) a következő hálót jeleníti meg:



Ez mit is ábrázol tulajdonképpen? A kék csomópontok jelentik a szabályt, azaz a veleszületett szívbetegséget. A piros csomópontok közül a legmagasabb szinten levők előteszteléseket végeznek (fentről az első kettő), majd a harmadik szinten levők a választ ellenőrzik. Kétféleképpen juthat el a szabályhoz a szabály jobb oldalán a tények közötti logikai kapcsolat miatt.

A háló pillanatok alatt bonyolultabbá válhat. Amikor eljutunk a betegséghez a következő képet kapjuk:



Amint látjuk, nagyon „sokszínű”. A bonyolultság viszont visszavezethető az attribútumok sokféleségére és számára. Minél több tesztet kell elvégeznie, annál elágazóbb és annál több csomópontra van szüksége.

Mivel egy szakértői rendszer csak akkor tekinthető közepes méretűnek, ha 500 szabálynál több van reprezentálva a tudásbázisában, az én szakértői rendszerem csak kis méretű. Az információ beszerzése, összegezése és szortírozása így is nem kis feladatnak bizonyult.

Az általam kiválasztott szakértő egy frissen végzett orvostan-hallgató volt. Feltettem neki a kérdést, hogy miket kérdez egy olyan betegtől, aki szívproblémákkal fordul hozzá. Ő is kihangsúlyozta a tényt, hogy ezen betegségek tünetei nagyon hasonlítanak, viszont vannak főtünetek, amelyek segítségével az orvosok is döntenek. Ezen kívül nem adhatunk meg egy teljes diagnózist, ha nem ismerjük a beteg leleteit, tehát nem adhatunk 100%-os megoldást. Viszont minden betegség esetén minden tünethez tartozik egy bizonyossági faktor, és így diagnosztizálható a betegség bizonyos valószínűséggel. Ha minden tünet megtalálható, amit a szakértő elmondott az adott betegséghez, természetesen a valószínűség nagy, ha csak néhány, akkor ez csökken.

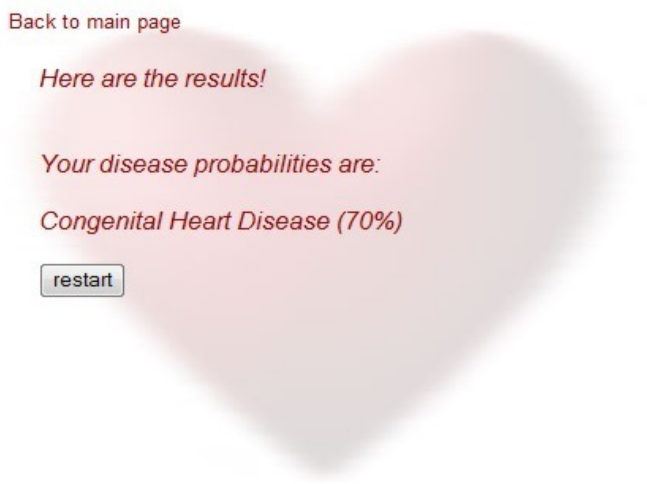
A tüneteket és a betegségeket ábrázoltam egy fa segítségével, és így könnyebb volt a szabályok reprezentálása. Heurisztikák alkalmazása pedig szinte elkerülhetetlennek bizonyult.

Nagyon fontos, hogy a tudásbázis áttekinthető legyen. Ez főként akkor mutat nagy jelentőséget, ha egy újabb betegséggel akarjuk bővíteni a rendszerünket. Így megkeressük a tüneteit, megnézzük, hogy az „ökölszabály” hogy alkalmazható, milyen családba tartozik, esetleg új

családdal bővítsük a rendszer tudását. Ez természetesen egyelőre jövőbeli terv, hiszen a tudásbázis összeállítása a nyolc betegség diagnosztizálására sem volt egyszerű.

A szakértői rendszerem szakterülete az az orvoslás, amely a szívre irányul. Bőven rendelkezik szakértőkkel, és közöttük nagy egyetértés van ami a betegségeket illeti. Nem csupán egy szakértőt kérdeztem meg, hanem egy orvostan-hallgatót is, aki szintén ugyanazokat a tünet-csoportokat sorolta fel az egyes betegségekhez. Ezen kívül meg a világhálón is rákerestem egy-egy betegség tüneteire, ha valami nem tűnt tisztának a szakértők elmondása alapján.

Az eredmény tehát egy olyan diagnosztizáló rendszer, amely tekinthető döntéstámogatónak (alkalmas például tanulásra is). Mégis a mindennapi, az orvosi szakterületben kevés tapasztalattal rendelkező emberek számára készült, akik aggályaikat feloszlathatják vagy megerősíthetik a rendszer használatával.



6. Összegzés

A szakértői rendszerek használatának sikerei azt mutatják, hogy van rájuk igény. Sokszor helyettesít egy igazi, emberi szakértőt, talán az orvostudományban való használata vetne fel némi etikai kérdést. De ez nem jelenti azt, hogy nem használhatják az emberek ha valami panaszuk van, hiszen ha megállapítja a betegséget, akármilyen kicsi valószínűséggel is, de ott van a betegség lehetősége és elmegy az illető az orvoshoz.

Napjainkban nagy számban használják a szakértői rendszereket, különböző területeken.

Egy kis múltbeli áttekintés végett:

Az Amerikai Egyesült Államokban 1985-ben 50, 1986-ban 350, 1987-ben 1100, 1988-ban 3200, 1992-ben 12500 alkalmazásról számoltak be. A mezőgazdaságban, üzleti életben, szállításban, iparban, orvosi területen, űrkutatásban egyaránt nagy számban használták és használják.

Európában a mesterséges intelligencia eszközök használatában Nagy-Britannia és Németország vezet, alig van lemaradva Franciaország. Közzétett adatok alapján 1988-1990 között 700 szakértői rendszer készült.

Napjainkban viszont már sokkal többet fejlesztenek és használnak. Szükség van rájuk, tehát a szakértői rendszerek jövője biztosítva van.

Irodalomjegyzék

Sántáné-Tóth Edit: *Tudásalapú technológia, szakértő rendszerek.* Dunaújvárosi Főiskola Kiadói Hivatala, 2000

Fekete István, Gregorics Tibor, Nagy Sára: *Bevezetés a mesterséges intelligenciába*, LSI 1990.

Dr. Bognár Katalin: *Mesterséges intelligencia* (egyetemi jegyzet)

Mesterséges intelligenciával és szakértői rendszerekkel kapcsolatos írások:

http://www.kbsc.com/whitePapers.html

http://www.kbsc.com/expertsys.html

Ernest Friedmann-Hill: *Jess in action* (Rule-Based System in Java)

A JESS hivatalos honlapja

http://jessrules.com

A JESS dokumentációja

http://www.jessrules.com/jess/docs/70