Debreceni Egyetem, Informatikai Kar
Számítógéptudományi Tanszék

Tudásalapú rendszer esettanulmány

Dr. Bognár Katalin
Tudományos főmunkatárs

Baka Zoltán
Programtervező matematikus

Debrecen, 2007

Köszönettel tartozom Dr. Bognár Katalinnak, diplomamunkám elkészítésében nyújtott segítségéért és Karácsony Sándornak, amiért megíratta velem életem első programsorát.

Tartalomjegyzék

Ábrák

Táblázatok

1. Bevezetés

1.1 A diplomamunka célja

Diplomamunkám célja egy olyan gyakorlatban is jól használható tudásalapú rendszer készítése, melynek megvalósításával jól szemléltetni lehet ezen rendszerek tervezését, implementálását és használatát, illetve a bennük rejlő lehetőségeket és természetesen korlátokat is. Maga az elkészített rendszer az AquAdvice nevet viseli. A végfelhasználó számára akvarisztikai problémák megoldására, illetve egy adott akváriumban uralkodó kémiai biológiai és egyéb környezeti folyamatok hatásainak tanulmányozására ad lehetőséget. Teszi mindezt úgy, hogy a felhasználónak minimális akvarisztikai ismeretekre van szüksége a rendszer használatához. A programmal történő interakció hasonló egy akvarisztikában jártas szakemberrel történő beszélgetésre: kérdésekből és a felhasználó által adott válaszokból áll. Az alapvető elképzelés a rendszer megalkotásakor az volt, hogy képes legyen a felhasználó akvarisztikával kapcsolatos napi teendőit segíteni, tanácsokat adni, hogy milyen feladatokat milyen sorrendben kell elvégeznie egy cél érdekében, illetve milyen eszközök szükségesek ezekhez. Ezzel a megoldással egy komplett folyamatirányítási szerettem volna készíteni.

Jómagam nemrég óta foglalkozok komolyan az akvarisztikával – de szinte az első pillanattól kezdve óriási érdeklődéssel és lelkesedéssel kezdtem tanulmányozni a hobbim megismeréséhez és hatékony űzéséhez szükséges információkat. Az interneten és szakkönyvekben fellelhető ismeretek és mások tapasztalatai világossá tették számomra, hogy ahhoz, hogy sikeresek legyünk a díszhaltartásban rengeteg ismeretet kell felhalmoznunk és napról napra a gyakorlatban is alkalmaznunk. Az akváriumban lezajló kémiai és biológiai folyamatok rendkívül összetettek, ugyanakkor szinte minden hal más-más igényekkel rendelkezik; mindez az akvarisztikával csak most ismerkedőket elrettentheti – vagy az általuk okozott csalódások a hobbi feladására kényszeríthetik a kezdő akvaristákat. Ezért döntöttem úgy, hogy meglévő tapasztalataimat felhasználva készítek egy olyan tudásalapú rendszert, mely segít megtenni az első lépéseket és megpróbálja felhívni a felhasználó figyelmét az esetleges problémaforrásokra ezzel elősegítve a hobbi sikeres űzését. Ugyanakkor reményeim

szerint a tapasztaltabb akvaristák is haszonnal, vagy legalábbis érdeklődéssel tekinthetnek a programra, amely a későbbiek folyamán egyszerűen továbbfejleszthető, még több ismerettel bővíthető.

Diplomamunkám második részében megpróbálok egy általánosabb képet adni a mesterséges intelligenciáról, miközben az AquAdvice rendszerre vonatkozó részeket bővebben is tárgyalom. A harmadik fejezetben megpróbálom megmutatni a tudásalapú rendszerek jellemzőit, illetve azok viszonyát a szakértő rendszerekhez, valamint részletezem, hogy milyen nagyobb lépésekből is áll egy ilyen rendszer fejlesztése. A negyedik fejezetben az AquAdvice rendszer tudásbeszerzése során használt PCPACK nevű rendszert elemzem, megmutatva azokat a főbb lehetőségeket, melyeket a tudásmérnökök számára biztosít, különös figyelemmel azokra, melyeket magam is felhasználtam a rendszer elkészítése közben. Az ötödik fejezetben az AquAdvice rendszert mutatom be, annak implementálásától kezdve a használatáig, illetve felvázolom, milyen lehetőségek vannak a rendszer továbbfejlesztésére.

2. A mesterséges intelligencia

2.1 Általánosságban a mesterséges intelligenciáról

2.1.1 A Dartmouth-i konferencia

A mesterséges intelligencia, mint tudományág körvonalai az 1956 nyarán Darthmouth College-ben tartott, John McCarthy által szervezett konferencián kezdtek kirajzolódni. Olyan nagy tudósok fektették le az MI alapjait McCarthy mellett, mint Allen Newell, Herbert Simon, Marvin Minsky, Nathaniel Rochester és Claude Shannon. Célkitűzésük az volt, hogy egy tíz tudósból álló csoport két hónap alatt választ adjon azon feltételezésre, mely szerint elviekben bármilyen intelligens viselkedést lehet olyan pontosan definiálni, mely leírás segítségével egy gép szimulálni tudja azt. Megpróbálták lefektetni a gépi tanulás, beszélés, problémamegoldás és sok más olyan probléma elméletének alapjait, melyeknek megoldása akkoriban az emberek sajátja volt. Azt gondolták, hogy a felvetett problémák egy részében jelentős előrelépést jelenthet, ha egy megfelelően kiválasztott tudóscsoport a nyár folyamán ezekre koncentrál.

2.1.2 Az intelligens viselkedés

A kezdeti cél az emberi gondolkodás számítógéppel való reprodukálása volt. A legáltalánosabban úgy fogalmazhatunk, hogy olyan eszközök (gépek, számítógépek, stb.) készítését tűzte ki céljának az MI, amelyek a külvilág felé intelligens viselkedést mutatnak.

A fenti körülírás nem definíció, hiszen az, hogy ki mit tekint intelligens viselkedésnek, erősen szubjektív. A mai napig nincs olyan mindenki által elfogadott definíció az intelligenciára, amelynek segítségével egyértelműen eldönthető egy viselkedésről, hogy az intelligens vagy sem. Még gyakran ugyanazon ember is egy adott viselkedést intuitíve intelligensnek tekint, míg a másik pillanatban már nem tekinti annak. Ebből következik, hogy a mesterséges intelligenciát is szinte lehetetlen egzakt módon definiálni és ebben a legtöbb MI kutató egyet

is ért. Ennek ellenére vannak többé-kevésbé használható és a lényeget megragadó megfogalmazások. A következő megfogalmazás az Encyclopedia Britannicában(2001) olvasható:

„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."

Minsky a következő gondolattal próbálta körülhatárolni a mesterséges intelligenciát, mint kutatási területet:

„Az a tudomány, melynek segítségével olyan cselekedetek végrehajtására teszünk képessé gépeket, melyek véghezvitele az embertől intelligenciát igényelne".

2.2 A Turing teszt és az MI

2.2.1 A Turing teszt

Valamilyen mértékben könnyíti a helyzetünket az ún. Turing-teszt. 1950-ben Alan Turing brit matematikus egy írásában egyértelműen kijelentette, hogy egy napon létezni fog olyan szerkezet, amely intelligenciája nem lesz megkülönböztethető az emberétől – tette mindezt úgy, hogy Gödelhez hasonlóan tisztában volt azzal, hogy már számos olyan számítási feladat van, amelyek gépekkel nem elvégezhető, nem megoldható – függetlenül az adott gép számítási sebességétől és tárolókapacitásától. Ezzel egy időben Turing megfogalmazott egy tesztet, annak eldöntésére, hogy egy adott számítógép képes-e erre. A tesztnek három egymástól elkülönített résztvevője van, három szobában: az intelligensnek feltételezett számítógép, egy a kérdésekre válaszoló ember és egy szintén ember kérdező. A teszt során a kérdező ugyanazokat a kérdéseket teszi fel mindkét résztvevőnek, de a teszt elején előtte nem ismert, melyik szobában ki (illetve mi) van. A kérdező a kérdéseket egy olyan eszköz segítségével teheti fel (pl. konzol), ami mindkét résztvevőnek egyenlő esélyeket garantál. A Turing-teszt akkor tekinthető sikeresnek, ha a kérdező nem tudja a válaszok alapján

egyértelműen eldönteni, hogy melyik szobában van a számítógép. Turing szerint ebben az esetben, minden okunk meglenne rá, hogy a komputert intelligensnek tekintsük.

2.2.2 Erős és gyenge MI

Megkülönböztethetünk erős és gyenge mesterséges intelligenciát; az erős mesterséges intelligencia kutatásának célja olyan konkrét programok elkészítése, amelyek ténylegesen úgy működnek, mint az emberi agy; a gyenge mesterséges intelligencia kutatások pedig elsősorban azokat az elveket, módszereket, és technikákat (illetve a már meglévőek fejlesztését) helyezik előtérbe, amelyek hasznosak, szükségesek ahhoz, hogy az emberi gondolkodást számítógépen reprodukáljuk.

A két különböző típusú MI-re a – nem szó szoros értelemben vett – definíciók a következők:

Gyenge MI: „A mesterséges intelligencia kutatásának célja, olyan gépeket alkotni, melyek azokat a feladatokat képesek elvégezni, amelyekben jelenleg az emberek jobbak náluk.” (Rich, Knight, 1990)

Erős MI: „Az erős MI támogatói úgy vélik, hogy amikor elkészül a megfelelő program, akkor, ha ezt egy számítógépen futtatjuk, tulajdonképpen egy tudatot (elmét) kapunk, és ekkor már nem lesz felfedezhető semmilyen különbség az emberi agy, és az ő viselkedését emuláló szoftver között.” (Russel, Norvig 2001 – FAQ Internet site)

A gyenge mesterséges intelligencia esetében csupán az intelligens viselkedés külső reprodukálásáról van szó, azaz csak az a cél, hogy „hihető” legyen a módszerrel megvalósított MI, tehát nem kell például a programnak ténylegesen úgy működnie - „gondolkoznia” - mint az emberi agynak, csupán annyi a cél, hogy a külső szemlélőnek ez úgy tűnjön. Ezzel szemben az erős MI kutatásának célja a tiszta intelligencián túlmenően kiterjed más emberibb vonások megértésére és számítógépen történő reprodukálására (pl. érzelmek, fáradtság, sőt akár öntudat).

2.2.3 A Turing tesz és az erős, illetve gyenge MI viszonya

Sok modern mesterséges intelligencia kutató úgy tekint a Turing-tesztre, mint egy óriási tévedésre, ami eltereli a kutatók figyelmét sokkal fontosabb kutatásokról, amelyek esetleg gyakorlatban is alkalmazható, hatékony eszközöket eredményeznének a célterületük egy-egy speciális problémájának megoldásához. Ahhoz ugyanis, hogy egy számítógép átmenjen a Turing-teszten – minden egyéb megszorítás és feltétel nélkül – erős MI-t kell fejleszteni, ami meglehetősen nehéz feladat. Ezek mellett, szükség van arra is, hogy olyan MI-t fejlesszünk, ami a teszt kérdezőjének minden olyan kérdésére is „emberszerűen" tud válaszolni, amelyre egyébként eléggé „gépies" választ adna – ezzel egyúttal le is leplezve magát. Vegyük ugyanis a következő példát: a kérdező egy eléggé összetett matematikai feladat eredményére kíváncsi; a gép feltehetőleg hamarabb válaszol – míg az emberi válaszadó lassabban és esetleg hibásan is. Egy másik szemléletes példa lehet, ha a kérdező arra kíváncsi, hogy a válaszadókban milyen hatást vált ki egy adott versrészlet; a gép feltehetően csak a szöveg szigorú elemzésének eredményével szolgálhatna, míg az emberben olyan mélyebb érzések is megjelenhetnének, amelyek nem következnek a szövegből. Az első eset kezelésére a gépnek olyan stratégiával kellene rendelkeznie, amely képes eldönteni azt, hogy egy ember milyen esetekben, milyen valószínűséggel hibázhat és körülbelül mennyi idő szükséges neki bizonyos kérdések megválaszolásához. Ez tulajdonképpen azt jelentené, hogy a programunkat nem determinisztikussá és lassúvá kellene tenni ahhoz, hogy átmenjen a Turing-teszten, ami nem feltétlen eredményezné a használhatóságát. A második esetben olyan plusz információkkal kellene ellátni a gépet, amelyek az ember érzelmeinek felépítésére, azok egymástól való függésére vonatkoznak – ez tekintettel arra, hogy az ember számára egy gép tényleges belső (nem a külvilág felé mutatott) érzései a gyakorlatban nem hasznosak, felesleges energia-befektetés lenne.

2.2.4 A Turing teszt módosítása

A fentiekből látszik, hogy a Turing-tesztet egy kissé módosítani kell ahhoz, hogy a gyakorlatban is használható verifikációs és validációs eszközt kapjunk eredményül. Véleményem szerint a tesztet úgy kell használni egy adott gép vizsgálatakor, hogy szigorú

szabályokat határozunk meg arra vonatkozóan, hogy a kérdező milyen jellegű kérdéseket tehet fel, sőt bizonyos esetekben még azt is meg kell szabnunk, hogy ezeket milyen formai követelményeknek megfelelően továbbíthatja a résztvevők felé. A továbbiakban, ha külön nem említem, az így módosított tesztet értem Turing-teszt alatt.

Ha az erős és gyenge MI „definíciókat" összevetjük a Turing-teszttel, semmi kétségünk nem lehet afelől, hogy mindkét típusú MI-hez tartozó számítógép (program) sikeresen végezné el azt. A gyenge MI-hez tartozó gép átmenne a teszten, mivel annak cselekedetei (válaszai) definíció szerint nem különböztethetőek meg az emberétől; az erős MI-hez tartozó gép pedig azért tenné le – feltehetően – sikeresen a tesztet, mert teljes mértékben az emberrel megegyező módon gondolkozik, azaz tulajdonképpen olyan mintha a tesztet két emberrel végeznénk, így értelemszerűen nem lehet egyértelműen eldönteni melyik válaszadó a számítógép. Ez utóbbinál viszont érdemes lehet megjegyezni, hogy – mivel más-más emberek gyakran ugyanarra a kérdésre is eltérő választ adnak – a tény, hogy a gép átmegy a teszten, nem feltétlenül azt jelenti, hogy válaszai megegyeznek az ember által adott válaszokkal, sőt egy ténylegesen gondolkodó számítógép adhatna olyan válaszokat is, melyeket önmagukban vizsgálva a gépet nem tekintenénk intelligensnek – akárcsak az embert egy másik, hasonló esetben.

2.2.5 Az erős MI nehézségei

A két típusú MI-t egy másik szemszögből is vizsgálhatjuk. Azt hiszem, hogy első pillantásra a legtöbben az erős MI-t tekintenék „jobbnak", ami nem is meglepő, hiszen ha az MI kutatás céljának az emberi agy lemásolását tekintjük, akkor ez a tökéletes megoldás. Erős MI készítése viszont még nem sikerült, és ha valaha sikerülni fog, feltehetőleg az rengetek pénzbe és időbe is fog kerülni – sőt, meg merem kockáztatni, hogy még morális problémák kereszttüzét is el kellene viselnie – addig, amíg gazdasági és egyéb szempontból megtérül a befektetés. Minden kétséget kizáróan hatalmas előrelépés lenne technológiai szempontból, viszont felvetődne a kérdés, hogy miért fektettünk ennyi energiát egy olyan szerkezetbe, ami tulajdonképpen teljes mértékben helyettesíthető egyetlen emberrel – hiszen annak agyát másolja le. Ez a futurisztikus lény bizonyos esetekben ugyanúgy unatkozna, elfáradna,

kedvtelenné válna, figyelmetlen lenne és hibázna – így esetleg nem végezné el a rábízott feladatot –, ahogy egy ember is teszi, ezáltal az érző és gondolkodó robot elveszítené azon előnyeit az emberrel szemben, amelyeket gép mivoltából nyert. Kézenfekvőnek látszana a probléma megoldására, hogy ezeket a valamilyen szempontból „negatívként" megítélt viselkedésmódokat, „ne programozzuk bele" a szerkezetünkbe. Ez a megoldás viszont – ha tényleg erős MI-t készítettünk – nem vezetne eredményre, mert valószínűleg előbb-utóbb megjelennének nála a nem kívánatos viselkedésmódok – magától megtanulná azokat; hiszen feltehetőleg egy csecsemő sem úgy jön a világra, hogy például nincs kedve dolgozni, vagy beszélgetni – ezek később válnak az eszköztárának részévé. Ugyanakkor korábban láthattuk, hogy ha olyan erős MI-t akarunk fejleszteni, amely kiállja a Turing-tesztet még sok más olyan dologra is figyelnünk kell, melyeknek gyakorlati haszna eleve megkérdőjelezhető, és szükségességük pusztán a teszt megfogalmazásából adódik.

2.2.6 A gyenge MI előnyei

Az előzőeket összefoglalva gyakorlati szempontból (legalábbis jelenleg) a gyenge MI a legfontosabb jelentőségű, azaz célunk inkább egy olyan gép készítése, ami a lehető legnagyobb mértékben a hasznunkra válik életünk egy adott területén. Ugyanakkor természetesen az erős MI kutatások is fontosak lehetnek és az eredmények nem csak a informatikában lehetnek hasznosak, hanem például a pszichológiában is.

A gyenge MI meghatározása viszont túlságosan is tág, ugyanis ha megnézzük, a meghatározásba beleillik egy egyszerű számológép is, hiszen megfelelő problémák esetén intelligens reakciókat mutat – holott feltehetően senki sem tekint mesterséges intelligenciának egy ilyen alkalmazást. Egy másik szemléletesebb példa az 1997-ben Gary Kasparov sakkmestert számos meccsen legyőző Deep Blue számítógép, melyet az IBM készített. Ezt a gépet, ha a Turing-tesztet csak sakkfeladványokkal végeznénk, valószínűleg egyértelműen embernek gondolnánk – tényleges emberi ellenfelének legnagyobb bánatára –, holott győzelmének oka egyedül az óriási számítási kapacitása volt, nem pedig az alak-felismerési, tanulási, fejlődési illetve következtetési képességei. Ezek alapján egyáltalán nem tekinthető intelligensnek (akárcsak egy számológép), viszont a gyenge MI jellemzői tökéletesen igazak

rá. A fentiekből látszik, hogy nem mindig könnyű eldönteni, hogy mi-melyik kategóriába tartozik, ha egyáltalán beletartozik valamelyikbe, de erre nem is feltétlenül van szükség.

2.3 Az MI segítségével megoldható problémák

Általánosságban kijelenthetjük, hogy azoknak a problémáknak a megoldásához nyújt segítséget a mesterséges intelligencia kutatás, amelyek megoldásánál a klasszikus informatikai eszközök csődöt mondanak, vagy korántsem olyan hatékonyak, hogy azokat a gyakorlatba átültetve is sikeresen használhassuk. A klasszikus informatika területéhez azon problémákat soroljuk, melyek megoldására algoritmus létezik. Itt viszont a „létezik" szó nem azt jelenti, hogy valaki már elkészítette a megfelelőt (persze ez sem kizáró ok), sokkal inkább azt, hogy elkészíthető egy olyan eljárás, amely hatékonyan megoldja az adott problémát. Itt hangsúly az elkészíthető és hatékony szavakon egyaránt van.

Előfordulhat, hogy egy probléma megoldásához elkészíthető egy algoritmus, ami megoldja azt, viszont olyan hatalmas mennyiségű tárhelyre és/vagy számítási kapacitásra lenne szükség a gyakorlatban való alkalmazásához, amely jelenleg – és a közeljövőben – nem áll rendelkezésünkre. Jó példa erre a sakkjáték. Mivel teljes információjú, kétszemélyes és véges játékról van szó, valamelyik fél számára létezik nyerő stratégia – egy olyan stratégia, amely minden a játék során előkerülő játékállás esetén megmondja, hogy mit kell lépnie. Ez a stratégia meghatározható a megadott – felépítésében nagyon egyszerű – algoritmus segítségével. Viszont tényleges meghatározása lehetetlen, mivel a játékfa exponenciálisan növekszik (kombinatorikus robbanás).

A másik változat, hogy az adott probléma megoldásához egyáltalán nem létezik algoritmus, vagy ha egyáltalán létezik is, nem tudjuk szavakba önteni, annak ellenére, hogy szinte naponta alkalmazzuk. Jó példa erre az, amikor beszélgetőpartnerünk arckifejezései alapján megállapítjuk, esetleg csak megsejtjük, hogy mit gondol például arról, amiről beszélgetünk, vagy éppen milyen a kedve – esetleg mindkettőt egyszerre. Könnyű belátni, hogy ezek a problémák algoritmikusan nem megoldhatóak, viszont a mesterséges intelligencia megpróbálhat valamilyen segítséget nyújtani a kezelésükhöz (pl. alakfelismerés segítségével).

2.3.1 Az MI korlátai

A mesterséges intelligencia területéhez tartozó módszerek többségére igaz az, hogy csodát nem várhatunk tőlük. Azaz – pont azért, mert az adott probléma nem algoritmikusan megoldható – a kapott eredményekről sohasem állíthatjuk azt, hogy száz százalékig megbízhatóak, ezért bizonyos alkalmazási területeken fenntartással kell kezelni őket. Egy számítógépes játékban a gépi ellenfél viselkedését irányító MI által javasolt cselekedeteknek nem valószínű, hogy komolyabb következményei lesznek – legfeljebb a gép elveszti a meccset; míg egy az iparban, űrkutatásban vagy orvostudományban alkalmazott MI rossz döntése komoly anyagi és esetleg emberi áldozatot követelhet, ha csak arra támaszkodunk.

2.4 Az MI vizsgálatának aspektusai

Az ember intelligens gondolkodását az MI több szemszögből próbálja közelíteni, mivel egy az egyben nagyon nehéz lenne lemásolni azt. Így a számításelméletből, kibernetikából, információelméletből, szimbolikus feldolgozásból és pszichológiából kialakuló tudományág folyamatosan több, egymástól jól megkülönböztethető (de nem teljesen független) terültre vált szét. Egyik része a kognitív következtetéssel foglalkozik és szorosan a matematikai logikához kapcsolódik. Egy másik terület sokkal inkább kapcsolódik az emberi érzékeléshez (hallás, látás stb.), ugyanakkor az előző terület eszköztárából is merít és alkalmanként bonyolult, formális eszközöket is használ (pl. alakfelismerés). A harmadik csoportot a szimbolikus MI alkotja, mely a szimbólumok segítségével megfogalmazott ismeretek feldolgozásával foglalkozik.

Az MI kutatók jelenleg ugyanúgy vizsgálják az emberi viselkedést, illetve feladatvégrehajtást, mint az emberi gondolkozást. Ezen két irányvonal mindegyikére igaz, hogy megtalálhatók bennük a tapasztalati (empirikus) illetve a racionális módszerekkel dolgozó elméletek.

Ezek alapján az MI a következő négy perspektívából vizsgálható [Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995]:

	Tapasztalati megközelítés	Racionális megközelítés
Emberi gondolkodás	Rendszerek, amelyek úgy gondolkodnak, mint az ember. Kognitív tudományok. *„Azon izgalmas próbálkozás, hogy számítógépeket gondolkozásra képessé teszünk...szó szerint egy gép öntudattal”.* *„Azon tevékenységek automatizálása, melyeket jelenleg az emberi gondolkozáshoz, értelemhez kapcsolunk, például döntéshozás, problémamegoldás, tanulás...”.*	Rendszerek, amelyek ésszerűen gondolkoznak. Matematikai logika. *„A mentális képességek tanulmányozása számítógépes modellek segítségével”.* *„Azon számítások tanulmányozása melyek lehetővé teszik az érzékelést, következtetést és cselekvést”.*
Emberi viselkedés	Rendszerek melyek úgy viselkednek, mint az emberek. Kognitív feladat-végrehajtás. *„Azon gépek készítésének művészete, melyek olyan funkciókat látnak el, melyek végzéséhez az embernek intelligenciára van szüksége”.* *„Annak tanulmányozása, miként tehetünk képesség számítógépeket olyan dolgokra, melyekben jelenleg az emberek jobbak”.*	Rendszerek melyek ésszerűen cselekednek. Következtető eljárások gyakorlati megvalósítása. *„Az a tudományterület, mely megpróbálja megmagyarázni, és szimulálni az intelligens viselkedést számítógépes módszerek segítségével”.* *„A számítástechnika azon területe, ami az intelligens viselkedés automatizálásával foglalkozik”.*

Táblázat 1: Az MI különböző perspektívákból

3. Szakértő rendszerek és Tudásalapú rendszerek

3.1 A tudásalapú rendszerek feladata

Egy tudásalapú rendszer feladatát tömören úgy fogalmazhatjuk meg, hogy egy adott területen előforduló probléma megoldására ad javaslatot, vagy annak okát próbálja kideríteni és azt is meg tudja mondani, hogy milyen módon jutott erre a megoldásra, hogyan következtetett a felhasználó által párbeszéd jellegű kommunikáció során megadott információkból a végeredményére – azaz magyarázó alrendszert is építettek bele. Ez utóbbi alrendszerre nem feltétlen van szükség, ahhoz hogy egy kiforrott rendszert hatékonyan alkalmazhassunk, viszont a rendszerfejlesztési, tesztelési illetve verifikációs szakaszában kimondottan hatékony eszköz lehet az esetleges hibák kiderítésére. Létezhetnek viszont olyan rendszerek is amelyekben elsődleges szerepet játszik ez az alrendszer, azaz a felhasználót nem is maga a végeredmény, hanem az ahhoz vezető következtetési lánc érdekli igazán. A tudásalapú rendszerekre általánosságban elmondható, hogy nem numerikus, hanem szimbolikus adatokkal dolgoznak.

A mesterséges intelligencia gazdaságilag is fontos kutatási területére remek példát nyújtanak a Tudásalapú Rendszerek (Knowledge-Based Systems – KBS). Tudásalapú rendszernek egy olyan alkalmazást (számítógépes programot, rendszert) tekintünk, amelyben az ismeretek a működés során használt következtető algoritmusoktól jól elkülönülő helyen, az ún. tudásbázisban tárolódnak. A következtetést megvalósító algoritmusokat általánosan fogalmazzuk meg, nem vesszük figyelembe, hogy konkrétan milyen adatokkal kell majd dolgozniuk. Ez a megvalósítás több célt is szolgál. Mivel egy-egy ilyen (a gyakorlatban is használt) rendszerben órási mennyiségű információval kell dolgozni, ha nem így tennénk, akkor adott esetben – pl. a rendszer evolúciójánál, esetleges hibajavításnál, stb. – szinte lehetetlen feladatnak bizonyulna az ismeretek és algoritmusok akár minimális mértékű módosítása. Másrészről a tudás ilyen tárolása lehetőséget ad arra, hogy a fejlesztési fázisban egyszerre csak egy – már önmagában is összetett – részfeladatra figyeljünk: egyszer a tudásreprezentációra, másszor a következtető rendszer implementálására.

3.2 A tudásalapú rendszerek és szakértő rendszerek egymáshoz való viszonya

A tudásalapú rendszerek részét alkotják a szakértő rendszerek (Expert Systems – ES). Sok angol nyelvű irodalomban szűkebb jelentést tulajdonítanak neki, mint a tudásalapú rendszereknek, míg más helyeken szinonimaként használják a két megnevezést. Azok a definíciók, melyek a szakértő rendszereket elkülönítik a tudásalapú rendszerektől, úgy fogalmaznak, hogy akkor beszélhetünk szakértő rendszerről, amikor a tudásalapú rendszerben tárolt ismeretek egy jól meghatározható speciális terület szakismeretét írják le; ezzel szemben a tudásalapú rendszereknél nincs ilyen „megszorítás", azaz ide olyan rendszereket sorolhatunk, melyek teljesen általános információkat dolgoznak fel. Példaként tekinthetjük egy orvos ismereteit – ha azok köré építjük fel rendszerünket, egy szakértő rendszert kapunk, mivel ezek az ismeretek eléggé kötődnek egy adott szakterülethez (orvoslás). Másik szemléletes példa lehet, ha a szabályos közúti közlekedéshez szükséges ismereteket próbáljuk meg rendszerünkbe integrálni – ekkor „csak" tudásalapú rendszerről beszélhetünk, mivel ezen ismereteknek feltehetően sok ember birtokában van, és egy részüket még azok is ismerhetik, akik nem rendelkeznek jogosítvánnyal. Számos forrásban megtalálható mindkét elnevezéssel illetett rendszer szerkezeti modellje, amelyek a legtöbb esetben teljesen megegyeznek. Ugyanakkor a két fenti példa jól szemléltet egy fontos eltérést a két rendszer között, mégpedig azok alkalmazási területét: az első rendszert valószínűleg, csak egy szűkebb területen fogják alkalmazni és feltehetően döntéseinek komolyabb következményei lehetnek, míg a második akár szélesebb körben is népszerűvé válhat például játékok formájában.

Felvetődik a kérdés, hogy akkor melyik a helyes megközelítés: az amely azonosnak tekinti a két rendszert, vagy az amely külön-külön említi őket (esetleg csak az egyik létezéséről beszél). Véleményem szerint mindegyik hozzáállás elfogadható, csak figyelembe kell vennünk, hogy mivel kapcsolatban szeretnénk nyilatkozni az MI ezen területéről. Ha pusztán technológiai megoldásokról, implementációról, stb. szeretnénk beszélni tökéletes megoldás a két rendszert egyként kezelni, ugyanis ezek a dolgok függetlenek attól, hogy mik is konkrétan a rendszerben tárolt ismeretek; a következtetési, magyarázó és más technikai jellegű algoritmusok felépítése egy az egyben megegyezhetnek a két rendszerben (annak ellenére, hogy természetesen ezek nem teljesen függetlenek a letárolt ismeretektől, legfőképpen azok

tárolási módjától). Másrészről, ha a rendszerünk alkalmazási területéről, a benne tárolt ismeretek jellegéről, illetve a hozzájuk szorosan kapcsolódó ismeret-beszerzési és tudásreprezentációs technikákról beszélünk, akkor hasznos lehet különválasztani a két dolgot.

3.3 Döntéshozó és döntéstámogató rendszerek

3.3.1 Döntéshozó rendszerek

A tudásalapú rendszereket két nagy csoportra oszthatjuk, annak alapján, hogy alkalmazási területén mennyire támaszkodnak, illetve egyáltalán milyen mértékben támaszkodhatnak az általuk szolgáltatott tanácsokra. Bizonyos rendszereknél a javasolt probléma-megoldási lépések minden további vizsgálat nélkül elfogadhatóak és kivitelezhetőek; ezek a döntéshozó rendszerek, melyek nem tapasztalt embereknek is engedélyezik a döntést olyan területeken is, melyek nem csak tapasztalatát, hanem szakképzettségét is meghaladják. Természetesen ezen rendszerek döntéseit is felülbírálhatja a felhasználó – ha szükségesnek érzi és felkészítettük erre azt –, viszont előfordulhatnak olyan speciális esetek, amelyekben nem tartanánk logikusnak egy döntéshozó rendszer használatát viszont nincs lehetőségünk – például idő hiányában – fölülbírálni a nem feltétlenül helyes döntéseket. Egyes szakirodalmakban a döntéstámogató rendszerek felhasználóit aktív felhasználóként említik, míg a döntéshozó rendszerek használóid passzív felhasználónak. Döntéshozó rendszerre jó példát adnak bizonyos számítógépes játékok, melyekben a gépi ellenfélnek hiányos ismeretei vannak az adott játékállásról és közel végtelen lépéskombináció áll rendelkezésére egy adott időpillanatban, így például nem alkalmazhatóak a kétszemélyes, véges és teljes információjú játékokban használható lépésjavasló algoritmusok.
Másik példaként megemlíthetjük az űrkutatásban alkalmazott tudásalapú rendszereket; ezeknél – mivel hatalmas anyagi kockázattal járnak – nem logikus döntéshozó rendszert alkalmazni, mégis ilyenekről beszélhetünk, mivel általában olyan eszközöket (például robotokat) kell irányítani, melyeknél létfontosságú az azonnali reakció egy-egy szituációra, viszont a kommunikációs akadályok (rádiójel terjedési sebessége) a közvetlen irányítást nem tennék lehetővé.

3.3.2 Döntéstámogató rendszerek

Más esetekben csak döntéstámogató rendszerről beszélhetünk, mivel bármennyire is tökéletesnek gondoljuk az elkészült alkalmazásunkat, bizonyos okokból kockázatos lenne feltétel nélkül megbízni benne, így ezek elsődleges feladata, hogy emlékeztesse a tapasztalt döntéshozót a lehetséges alternatívákra, felhívja a figyelmét bizonyos következményekre – tehát valamilyen mértékben segítsen visszaemlékeznie olyan dolgokra, melyeket esetleg elfelejtett. Tipikus példát a döntéstámogató rendszerekre az orvostudományban alkalmazott diagnosztikai szakértő rendszerek adhatnak; ezen a területen feltehetően nem lesznek olyan rendszerek, amelyeket döntéshozóként fognak alkalmazni, még akkor sem, ha esetleg a távoli jövőben elképzelhető egy, az erős MI kategóriába sorolható Szakértői Rendszer, mely rendelkezik egy orvos minden tudásával, illetve gyakorlati tapasztalatával. Egy ilyen „lény" alkalmazása a tényleges orvoslásban nyilvánvalóan etikai problémákat vetne fel.

3.4 Mikor van szükség tudásalapú rendszerre?

3.4.1 Fontosabb szempontok

Ezek után felvetődik a kérdés, hogy mikor is célszerű a klasszikus informatikával szemben inkább a mesterséges intelligenciát, azon belül is a tudásalapú rendszereket előnyben részesíteni, ha egy problémát szeretnénk megoldani. Ez a következő esetekben javasolt:

- A probléma, amit meg kell oldanunk nagyon bonyolult, nehéz átlátni a feladatban az összefüggéseket.
- Egyszerre több ember ismeretére, szakértelmére van szükség, viszont ezek egy időben személyesen nem elérhetőek, vagy kifejezetten kevés ilyen szakértő van.
- Olyan specifikus ismeretekre van szükség, amelyhez csak nagyon kevesen értenek, viszont gyakran szükség van az ehhez a területhez tartozó problémák megoldására, és nem mindenhol érhető el emberi szakértő.
- Az embernél gyorsabb válaszadásra van szükség, például olyan esetekben, melyeknél a késői reagálás nagy kockázattal járna.

- Nagy mennyiségű információval kel dolgozni, melyek strukturálatlan formában állnak rendelkezésünkre.
- Hiányos ismeretek mellett is szükség van döntésre.

A fentiek mellett még egy további előnyt is eredményez, ha tudásalapú rendszert fejlesztünk: megőrizzük a szakértelmet, finomítjuk azt és mások számára is elérhetővé tesszük; azaz az ember tudásával ellentétben egy ilyen rendszer az idők folyamán nem veszít képességeiből. A fenti felsorolásból látható, hogy az egyik legfontosabb dolog, hogy ilyen alkalmazásokra leginkább abban az esetben lehet szükség, ha valamilyen oknál fogva a problémát ismerő és azt megoldani képes szakember nem elérhető. Ennek oka lehet az, hogy egyáltalán nincs ilyen szakember, vagy az, hogy túlságosan sok helyen lenne szükség a segítségére. A tudásalapú rendszerekkel szemben több hátránnyal is indulnak az emberi szakértők: elfáradnak (mentálisan és fizikálisan is), fontos részletekről feledkezhetnek meg, nem mindig következetesek, lassabban dolgoznak nagy mennyiségű adattal, memóriájuk sebessége és kapacitása is erősen korlátolt. Ezenkívül, még azzal is számolnunk kell, hogy bizonyos szituációkban az emberek hajlamosak elhallgatni, elferdíteni dolgokat, ha személyes érdekeik úgy kívánják; mindezek mellett az egyik nagy hátrányuk hogy halandóak, azaz szakértelmüket csak életükben tudják hasznosítani – hacsak nem készült egy alkalmazás az ő ismeretükre alapulva.

3.4.1 A tudásalapú rendszerek hátrányai

Ugyanakkor a tudásalapú rendszerek rendelkeznek néhány hátránnyal is, melyek közül a legfontosabbak a következők:

- Józanész hiánya: egy komputer számára semmi sem triviális; előfordulhat, hogy a nem megfelelő (nem eléggé alapos) tudásbeszerzés miatt, olyan következtetésre jut a rendszer, amely teljesen abszurd. Nagy rendszernél hiába fektettük óriási energiákat a fejlesztésbe és tudásbeszerzésbe, sohasem lehetünk teljesen biztosak abban, hogy az alkalmazás nem fog olyan eredménnyel szolgálni, amit az emberi szakértő józan esze biztosan kizárna.

- Alkalmazkodóképesség hiánya: Az emberi szakértők rugalmasan kezelnek új, addig nem tapasztalt szituációkat, alkalmazkodnak hozzájuk és megtanulják kezelni őket. Ezzel szemben, egy számítógépes program csak minimálisan mutatja ezeket a tulajdonságokat.
- Természetes nyelv: az emberi szakértővel sokkal kényelmesebben lehet konzultálni, sokkal részletesebb információkkal szolgálhat a felhasználónak. Ezzel szemben a számítógépes rendszereknek sokat kell fejlődni ezen a téren.

3.5 A tudásalapú rendszerek felépítése

3.5.1 A rendszer fejlesztői

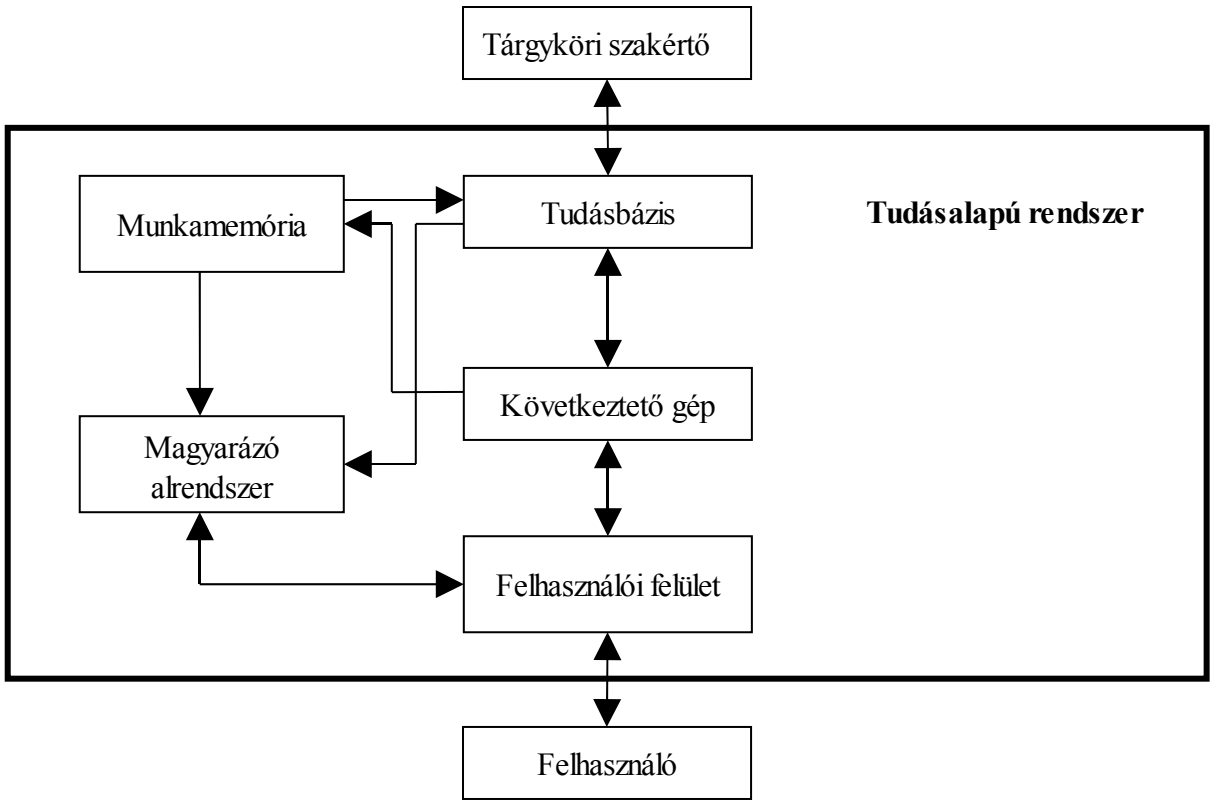
Egy tudásalapú rendszer elkészítésében két kulcsfigurát feltétlenül meg kell említenünk. Az egyik a tárgyköri szakértő, a másik pedig a tudásmérnök. A tárgyköri szakértő többnyire nem rendelkezik informatikai ismeretekkel, így a rendszer implementációjában nem vesz részt. Annál fontosabb feladat hárul rá a rendszerben használni kívánt ismeretek összegyűjtésénél: ő az akinek ismereteit, szakértelmét megpróbáljuk egy alkalmazásba sűríteni; ez a folyamat a tudásbeszerzés. A feladata azonban ismereteinek átadása után nem szűnik meg, hiszen a rendszer validálása szintén az ő hatáskörébe tartozik. Ez a folyamat azt jelenti, hogy megvizsgáljuk az elkészült szoftvert, hogy megfelel-e a korábban részletesen lefektetett követelményeknek, azaz tényleg azt a szoftvert készítettük-e el, amit szerettünk volna. Ebben a fázisban viszont nem pusztán a rendszer „lelkét” vizsgáltatjuk a tárgyköri szakértővel, hanem a felhasználói felületet, a rendszer kezelhetőségét, a felhasználói dokumentációk megfelelőségét stb. Természetesen ebbe a fázisba érdemes bevonni a szoftver célközönségét is, azaz azokat a tárgyköri szakértelemmel nem feltétlen rendelkező egyéneket is, akik később a rendszert használni fogják. Mint korábban már szó volt róla, egy tudásalapú rendszer nem csak egy ember, egy szakértő ismereteinek eltárolására szolgálhat, hanem több ember együttes tudását is ötvözheti. Ebben az esetben viszont a tudásbeszerzés során előfordulhatnak olyan problémák, melyekben a két (vagy több) tárgyköri szakértő nem ért egyet. A másik kulcsfigura a tudásmérnök. Ő az, aki informatikai tudását felhasználva a tárgyköri szakértő ismereteit formalizálja, rendszerezi majd a megfelelő fejlesztő eszköz kiválasztása után,

segítségükkel elkészíti a rendszert. Természetesen nagyobb rendszereknél több tudásmérnök dolgozik együttesen, részfeladatokra bontva a program elkészítésének lépéseit. Ha a rendszerünk elkészült és sikeres volt a validáció, akkor a végfelhasználó kapja kézhez és megkezdődik a rendszer használata. Ebben a stádiumban derülnek ki a leggyakrabban hibák, így ahogy korábban említettük érdemes a felhasználókat bevonni a validálás folyamatába, melynek leggyakoribb eszköze a tesztelés.

3.5.2 A rendszer főbb részei

A tudásalapú rendszerek legfontosabb részei a tudásbázis, a munkamemória és a következtető gép. Mint korábban is említettük, a tudásbázisban jól elkülönítve található meg a rendszer ismeretét jelentő adathalmaz (tények, következtetéshez használható eszközök), mely felépítése függ a rendszerünkben alkalmazott ismeretreprezentációs módszertől. A három legelterjedtebb módszer a szabályalapú, keretalapú és esetalapú reprezentáció. A munkamemória az a feladat-specifikus terület, ahol olyan információkat tárolunk átmenetileg, amelyek az éppen aktuális probléma megoldásához szükségesek. Az egész rendszer lelkét a következtető gép adja. Ez egy olyan – az ismeretreprezentációs módszertől nagyban függő – mechanizmus, mely az ismeretbázisban található tényekből és a felhasználó által megadott probléma-specifikus állításokból, megpróbál valamilyen következtetésre jutni a tudásbázisban található következtetési eszközök segítségével.

Természetesen egy tudásalapú rendszer a fenti legfontosabb részeken kívül még számos alrendszerrel rendelkezik (rendelkezhet). A magyarázó alrendszer nem mindig kerül implementálásra. Ennek feladata, hogy a felhasználó számára érthető módon megjelenítse a végeredményhez vezető következtetés lépéseit, így lehetőséget adva (például egy döntéstámogató rendszernél) arra, hogy megfontolja a program által adott választ. Másrészről, a rendszer fejlesztési szakaszában is nagy segítséget nyújthat a tudásmérnöknek és a szakértőnek a szemantikai hibák felfedezésében.



Ábra 1: Egy tudásalapú rendszer felépítése

3.6 Tudásbeszerzés

3.6.1 A tudásbeszerzés célja, feladatai

A tudásbeszerzés talán az egész rendszer fejlesztésének egyik legfontosabb részfeladata. Azt a folyamatot jelenti, melynek során különböző technikákat alkalmazva a tudásmérnök megszerzi a tárgyköri szakértőtől azokat az információkat, melyekből a rendszer alapját képező tudásbázist fel fogja építeni. Segítésére számos eszközt fejlesztettek ki, de ezeknek tulajdonképpen a lényegen nem változtatnak, csak megkönnyítik a tudásmérnök dolgát. A tudásbeszerzés több különböző technikával is végezhető, melyek mindegyike megpróbálja orvosolni a következő problémákat, melyek felmerülnek a folyamat során:

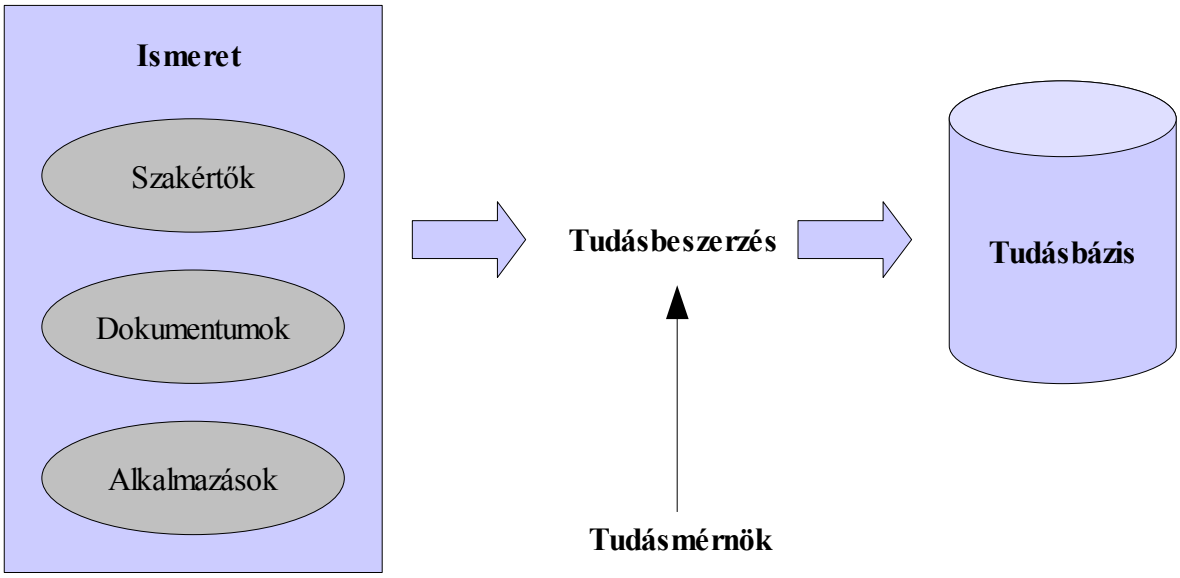
- a legtöbbször a tudás a szakértők fejében van, nem pedig kézzelfogható forrásokban
- a szakértők óriási mennyiségű információval rendelkeznek a szakterületükön

- problémát jelent a szakértők „passzív tudása”, ami a következőket jelenti:
  - nincsenek tisztában azzal, hogy milyen ismereteket birtokolnak és használnak
  - a passzív tudást nagyon nehéz megragadni (ha nem lehetetlen)
- az igazi szakértők nehezen érhetőek el, általában elfoglalt emberek
- nincs olyan szakértő, aki mindent tudna az adott területről
- a tudásuk nem feltétlen örök érvényű
- előfordulhat, hogy két ugyanazon területen dolgozó szakértőnek más-más a véleménye egy adott problémáról

Ezek fényében olyan tudásbeszerzési technikákra van szüksége, melyek:

- rövid időre rabolják el a szakértőket a munkájuktól
- lehetővé teszik, hogy a laikusok számára is érthető módon fogalmazzuk meg a tudást
- csak a rendszer szempontjából lényeges ismeretekre fókuszálnak
- lehetővé teszik a passzív tudás megragadását is
- lehetővé teszik, hogy a tudást több különböző szakértőtől is beszerezhessük
- lehetővé teszik a tudás ellenőrzését és bővítését a későbbiekben

Bár a megszerzett tudás forrása legtöbbször a tárgyköri szakértő, gyakran előfordulnak olyan esetek, amikor a tudásmérnöknek szakértő nélkül kell dolgoznia. Ennek oka lehet például, hogy a szakértő nem elérhető, de az is lehetséges, hogy a tudásmérnök olyan plusz ismeretekkel szeretné a rendszer felruházni, melyek nincsenek az adott szakértő birtokában. Ekkor különböző szakirodalmakhoz fordulhat, majd azok elemzésével bővítheti a tudásbázist. Ennek egy feltétele azonban, hogy már rendelkezzen olyan alapismeretekkel az adott szakterületen, hogy felismerje annak kulcsfontosságú fogalmait, összefüggéseit. A 2-es ábra a tudásbeszerzés lényegét szemlélteti.



Ábra 2: Tudásbeszerzés

Számos módszer alakult ki a tudásbeszerzésre, ami a szakértői ismeretek széles skálájának köszönhető. Más-más technikák alkalmazására van szükségünk, ha az aktív ismeretekre és ha a passzív ismeretekre vagyunk kíváncsiak. Ugyanígy különböző szakterületek ismereteit más-más módszerek ragadják meg könnyebben.

3.6.2 Kapcsolatháló

A kapcsolatháló egy olyan gráf, melynek csomópontjaiban az adott szakterület fogalmai vannak. A tárgyköri szakértőnek fel kell rajzolnia az egyes csomópontok közötti kapcsolatokat szimbolizáló nyilakat. A nyilakat fel kell címkéznie, annak megfelelően, hogy milyen kapcsolatban áll a két elem egymással. Nagyon jól alkalmazható az alapismeretek megragadására, a tudásbeszerzés első felében.

3.6.3 Folyamatábra

Olyan diagramm, melynek segítségével a szakterületen előforduló folyamatokat lehet megjeleníteni. Megtalálhatóak benne az egyes folyamatban résztvevő elemek, illetve folyamatok által előállított végtermékek is. Jól használható annak ábrázolására, hogy milyen

módon kerülnek alkalmazásra az egyes ismeretek a gyakorlatban. Készítheti a tárgyköri szakértő is, vagy akár tudásmérnök is. Utóbbi esetben ez a technika alkalmasabb lehet a passzív tudás feltérképezésére. Ehhez a módszerhez hasonló az állapot diagram is, mellyel a különböző szakterületi folyamatokban résztvevő objektumok különböző állapotait, illetve az azokból elérhető újabb állapotokat ábrázolhatjuk.

3.6.4 Interjú

Az egyik legáltalánosabb technika az interjú. Ennek során a tudásmérnök kérdéseket tesz fel a rendszer által lefedni kívánt témakörben egy ezen szakterületen jártas szakértőnek, aki azokra válaszol. A kérdésekre adott válaszokban rejlő ismereteket a tudás mérnök megpróbálja mérnöki módon megragadni, illetve rendszerezni azokat. Általában hangfelvételt készítenek a beszélgetésről, és csak később dolgozzák fel azt. Három típusról beszélhetünk, melyek mindegyike fontos szerepet játszik egy tudásbeszerzési folyamatban.

Strukturálatlan, spontán interjú: nagy szabadságot enged meg mindkét félnek; célja, hogy tisztázza a felhasználni kívánt ismeretek határait, illetve kiindulópontként szolgáljon a későbbi interjúk számára.

Félig strukturált interjú: a tudásmérnök a megelőző interjúk alapján előre felkészül, és olyan kérdéseket fogalmaz meg, melyek nagyban elősegíthetik a tudásbázis megalkotását. Ugyanakkor ez a fajta interjú típus lehetővé teszi, hogy miközben válaszokat ad, a tudásmérnökben felmerülő, újabb pontosítást igénylő kérdéseket is megválaszolja a szakértő. Gyakran a kidolgozott kérdéseket előre elküldik a szakértőknek, hogy felkészülhessenek azokra. Egy-egy ilyen interjú általában 1 órát vesz igénybe. Az egyik legkedveltebb típus, mivel segíti, hogy a szakértő a lényegre fókuszáljon, és csak kulcsfontosságú információkat szolgáltasson a tudásmérnök számára.

Strukturált, kötött interjú: semmiféle rugalmasságot nem enged meg a tudásmérnök számára, aki ezt a már jó előre megfogalmazott kérdéseivel éri el. A legtöbb esetben egy ilyen interjú inkább táblázatok kitöltésével és diagramok rajzolásával telik semmint beszélgetéssel.

A fent említett interjú technikák egyik előnye, hogy a tudásbeszerzés teljes folyamatában sikerrel használhatóak, azaz a kezdeti alapismeretektől egészen a legmagasabb szintű bonyolult ismeretek beszerzéséig alkalmas eszközként szolgálnak a tudásmérnök számára. Nagy hátrányuk viszont, hogy csak azt a tudást, ismeretet tudják megragadni, amivel maga a szakértő is tisztában van. Sok szakterületnél, viszont kiemelten fontos szerepet játszhat az óriási passzív ismeret, amely sokkal kevésbé kézzelfogható, mint a szakértő explicit tudása. Annak érdekében, hogy az ilyen területeken is eredményesen dolgozhasson a tudásmérnök, kiegészítő technikákra van szüksége.

3.6.5 Kommentálás

Az egyik legegyszerűbb technika a kommentálás. Nagyban hasonlít az interjúhoz, viszont egy fontos különbség van: a tudásmérnök csak passzív megfigyelőként vesz részt a folyamatban, nem kérdez a szakértőtől. Maga a tárgyköri szakértő több, az adott területen előforduló rutinfeladatot végez el, problémákat old meg, mindezt pedig teszi úgy, hogy folyamatosan kommentálja a cselekedeteit, döntéseit. Néha viszont a beszéd akadályozhatja a szakértőt, elvonhatja a figyelmét a feladatról. Ezért gyakran csak utólag szokta a szakértő (vagy egy másik szakember) kommentálni a cselekedeteit mialatt egy videó-felvételről visszanézik az egész folyamatot. Nagy előnye ennek a módszernek, hogy passzív ismereteket is előhozhat, melyek csak konkrét feladatok elvégzése közben kerülnek felszínre. Hátránya viszont, hogy nem alkalmazható a teljes ismeret-beszerzési folyamatban, inkább annak csak a végső fázisában, amikor a tudásmérnök már rendelkezik elég alapismerettel, ahhoz, hogy hasznosíthassa a kapott információkat.

3.6.6 Visszajelzés

A tudásmérnök összefoglalja saját szavaival azokat az ismereteket, melyeket korábbi interjúk (vagy akár más tudásbeszerzési módszerek) alkalmával szerzett. A tárgyköri szakértő ezután, megjegyzéseket fűz a hallottakhoz, hogy kiküszöbölje a hibákat és esetleges félreértéseket. Elképzelhető olyan változata is melynél a tudásmérnök más forrásokból nyert információkról kéri ki a szakértő véleményét.

3.6.7 Jegyzőkönyvelemzés

Egy másik hatékony módszere a szakértői tudás megragadásának az egyes adott szakterületen végbemenő folyamatokról készített jegyzőkönyvek elemzése. Ezt az eljárást gyakran az interjúkról készített jegyzőkönyvön is elvégzi a tudásmérnök. Ennek eredményeképpen a legtöbbször alapvető ismeretekre, alapfogalmakra, tulajdonságokra, értékekre, feladatokra és a köztük lévő kapcsolatokra deríthetünk fényt. Gyakorlatban ez úgy történik, hogy például egy interjúból készített jegyzőkönyvben, különböző színekkel kiemeljük a témához kapcsolódó kulcsszavakat, ügyelve arra, hogy a színekkel a különböző csoportokat jelezzük. Bizonyos esetekben előfordulhat, hogy az egyes tünetek (tények), hipotézisek (feltevések) és a diagnózisok kiemelése jelenti a folyamatot.

3.6.8 Laddering technikák

A Laddering („lépcsőző”) technikák lényege, hogy minden esetben hierarchikus, egymásra épülő ismereteket próbál megragadni. A technika eredményeképpen szinte csak ábrákat kapunk, melyek legtöbbször egy-egy fastruktúrát ábrázolnak, ahol a fa egyes elemeiben a szakterület kulcsszavai kapnak helyet. Az ilyen fastruktúrát a tudásmérnök és a tárgyköri szakértő közösen készíti el vagy papíron, vagy számítógépen. A folyamat során bármelyik résztvevő módosíthat, hozzáadhat, törölhet, átnevezhet vagy akár áthelyezhet adatokat az ábrában. Az ilyen technikákra jellemző kérdések lehetnek a következők:

- „Meg tudná nevezni néhány altípusát ennek?”
- „Miből állapíthatjuk meg, hogy ez a valami rendelkezik azzal a tulajdonsággal?”
- „Milyen részekből áll ez?”

3.6.9 Mátrik alapú technikák

A mátrix alapú technikák legtöbbször kétdimenziós táblázatok készítését jelentik, melyek a következő információk közötti viszonyokat tartalmazzák strukturált formában:

- Tulajdonságok – Értékek
- Problémák – Megoldások
- Hipotézisek – A diagnózishoz szükséges technikák
- Feladatok – Szükséges erőforrások

Maguk a mátrix elemei lehetnek akár szimbólumok (kérdőjel, pluszjel, mínuszjel stb.), színek, számok vagy leggyakrabban szöveges információk. Az ilyen technikákat jellemzően a korábban szerzett ismeretek ellenőrzésére, validálására és rendszerezésére használhatjuk, semmint újabb ismeretek beszerzésére. Ebből kifolyólag a technikát csak más módszerekkel kiegészítve használhatjuk.

3.6.10 Válogatás, osztályozás

Ezek a módszerek remek eszközök arra, hogy megtudjuk a szakértőről, milyen módon hasonlítja össze a különböző tulajdonságokat, és hogyan rendszerezi az ismereteit. Eredménye legtöbbször a különböző osztályok, tulajdonságok és azok prioritásai lehetnek.
Legegyszerűbb formája a kártyaválogatás. Ekkor a szakértő kártyalapokat kap, melyek mindegyikére egy-egy objektum nevét írtuk. A feladata az, hogy olyan csoportokat alakítson ki, melyek a bennük szereplő kártyák valamilyen közös tulajdonságát hivatottak jelezni. Mindezt akár többször előről kezdve, minden egyes alkalommal megadva az adott kártya-halomban szereplő lapok közös ismertetőjegyét.

3.6.11 A három kártyás trükk

Ez a technika sokban hasonlít a válogatásos és osztályozásos módszerekhez. Itt három véletlenszerűen kiválasztott objektumot ábrázoló kártyalapot kap a szakértő. Ezután, meg kell

állapítania, hogy miben hasonlít két objektum egymásra, és miben különböznek a harmadiktól. Ez a módszer tökéletes arra, hogy az objektumhoz olyan tulajdonságot rendeljünk, mely nem magától értetődő és a szakértő sem feltétlen gondol rá, amikor például interjú technikával kérjük, hogy jellemezze az adott objektumot. Ebből kifolyólag, a három kártyás trükk tökéletes módszer a passzív tudás megragadására.

3.6.12 Időben, információban korlátolt technikák

Ezeknek a tudásbeszerzési módszereknek a közös jellemzőjük, hogy csak előre meghatározott, rövid ideig rabolják a tárgyköri szakértő idejét. Adott egy konkrét időintervallum, amely rendelkezésére áll, hogy véghezvigyen egy olyan feladatot, melynek elvégzéséhez általában jóval több időre van szüksége. Ennek egy változata, amikor a szakértő rendelkezésére bocsátott információk is hiányosak. A tudásmérnök mindeközben elemzi a szakértő tevékenységét. Ezzel a technikával gyorsan és hatékonyan megállapíthatjuk, melyek azok a kulcsfontosságú részfeladatok és információk melyeket az adott szakértő elsődleges fontosságúnak tart az adott probléma megoldásában.

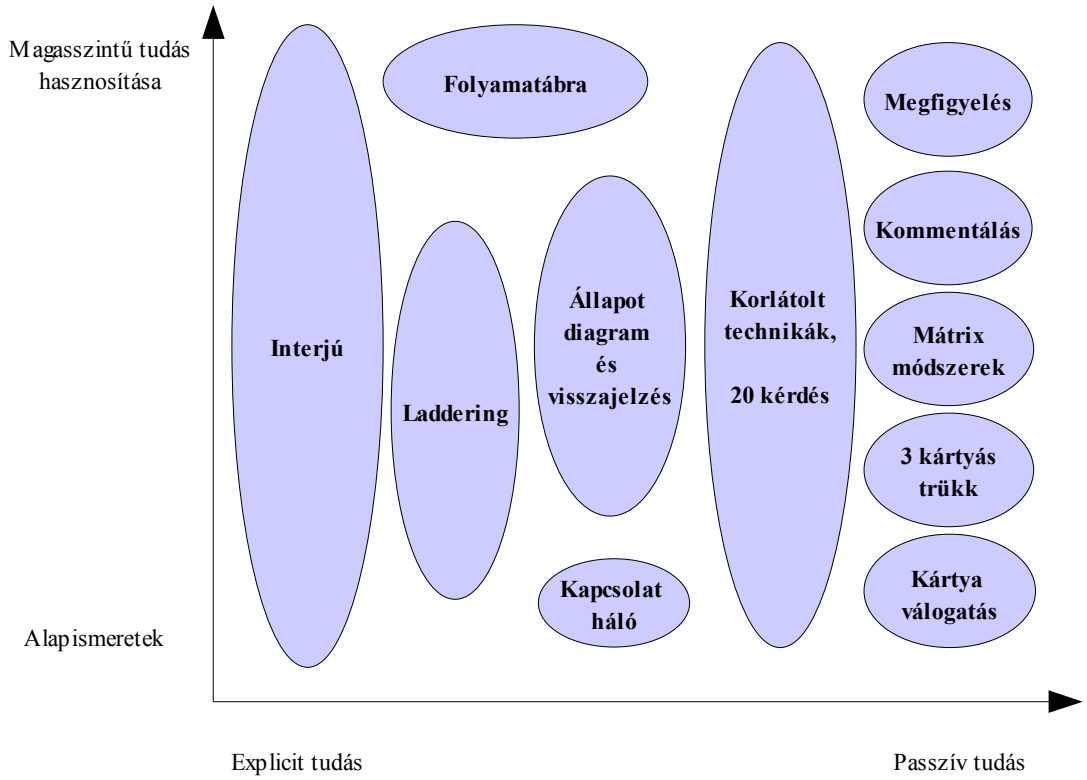
3.6.13 A 20 kérdéses technika

Az időben és információban korlátolt technikák egy érdekes változata a 20 kérdéses technika. A cél, hogy a szakértő kitalálja, mire gondol a tudásmérnök. Csak olyan kérdéseket tehet fel, amelyekre „igen”-nel vagy „nem”-mel lehet válaszolni. A tudásmérnök mindeközben rögzíti a szakértő kérdéseit. Maguk a kérdések, illetve azok feltevésének sorrendje a fontos tulajdonságok prioritásának sorrendjét fedheti fel.

3.6.14 A különböző módszerek összehasonlítása

A fentiekben láthattuk, milyen sokféle módszerrel lehet a szakértő tudását megragadni, előcsalni illetve konkretizálni. Mindegyik módszer különbözik a másiktól, mégpedig azért, mert az általuk beszerzett ismeretek, azok hasznosítása illetve a tárolásuk a szakértő emlékezetében is különböznek egymástól. Egyetlen módszerrel biztos, hogy nem hozhatunk

létre megfelelő tudásbázist, melynek a következménye a teljes rendszer használhatatlansága lenne, hiszen ez képezi annak az alapját. Mindenképpen szükségünk van a technikák kombinálására, esetleg még igazítanunk is kell majd őket az adott szakértő, illetve a szakterület speciális igényeihez. Ennek megkönnyítését szolgálhatja a következő ábra, melyben az egyes említett technikák különböző ismeretfajták kiderítésében való használhatóságát ábrázoltam.



Ábra 3: Tudásbeszerzési technikák

A fenti ábrából jól látszik, hogy melyik módszert milyen tudásnál, illetve a tudásbeszerzés kezdeti vagy végső folyamatában érdemes használni. Például a megfigyelés hatékonyan alkalmazható a passzív tudás feltárásához, viszont inkább a végső fázisban, amikor már a tudásmérnök elég ismerettel rendelkezik a szakterülettel kapcsolatban, ahhoz, hogy meg is értse azt, amit lát. Fontos még megjegyezni, hogy ahogyan haladunk az ábrán a jobb felső sarok irányába, a technikák egyre több idejét és energiáját emésztik fel a tudásmérnöknek és gyakran a tárgyköri szakértőnek is.

3.6.15 Egy tudásbeszerzés folyamata

Nagyon fontos a megfelelő tudásbeszerzési eljárások kiválasztása és azok megfelelő kombinálása, hiszen ez nagyban befolyásolhatja az egész rendszer fejlesztésének sikerességét. A következőkben egy tipikus tudásbeszerzési folyamatot mutatok be, melynek tanulmányozásával, könnyen kialakíthatjuk a saját tudásalapú rendszerünk fejlesztéséhez használt stratégiánkat. A tudásbeszerzést először az egyszerűbb módszerekkel kezdjük és fokozatosan haladunk az összetettebb technikák felé.

- Strukturálatlan, spontán interjú a szakértővel, annak érdekében, hogy:
  a) kiderüljön, milyen tudást kell beszerezni
  b) megtudjuk milyen célja lesz a tudás eltárolásának
  c) megállapítsuk melyek a szakterület az alapfogalmai
  d) megtaláljuk a közös nevezőt a szakértővel

- Az előbb elkészített interjú anyagok elemzése. Laddering technikák segítségével egy tömör, összefoglaló jellegű ábra készítése, mely vázlatosan ábrázolja az adott szakterület ismereteit. Olyan kérdések készítése az előbb elkészült ábrák alapján, melyek megválaszolásával a szakértő tisztázhatja a felmerülő ellentmondásokat.

- Félig strukturált interjú a szakértővel, az elkészült kérdések felhasználásával.

- Az interjú anyag elemzése, kulcsfontosságú alapismeretek megállapítása; tulajdonságok, az általuk felvehető értékek, objektumok közötti kapcsolatok, elvégzendő feladatok és szabályok feltérképezése.

- A megszerzett tudás ábrázolása a neki leginkább megfelelő technikákkal, például lépcsőző technika, mátrixok, hálózati diagramm, szövegkiemelés stb. Kiegészítésképpen strukturált dokumentumokat is készíthetünk, melyek valamivel részletesebben (szöveges formában) közvetítik a megszerzett ismereteket.

- Az eredmények feldolgozása a szakértő bevonásával lépcsőző technikákkal, hangos gondolkozással, húszkérdéses módszerrel, kapcsolatháló és állapotdiagramok készítésével – eredményül egyre részletesebb ismereteket szerezve a szakterületről. Itt már szinte bármely, korábban említett tudásbeszerzési eljárás (kommentálás, megfigyelés, kártyaválogatás, háromkártyás trükk, stb.) alkalmazható, mivel már kialakult egy olyan alap, amelyre építkezni lehet.

- A fenti eljárássorozat ismétlése egészen addig, amíg mind a tudásmérnök, mind a szakértő elégedett az eredménnyel, azaz kialakult az ismeretbázis, melynek felhasználásával tovább lehet lépni az alkalmazás fejlesztésének következő lépcsőfokára.

- Az ismeretek validálása más tárgyköri szakértők bevonásával.

3.7 Tudásreprezentáció és következtetés

3.7.1 Elvárások

A tudásreprezentáció szinte elválaszthatatlan a szakértő rendszerünkben alkalmazott következtetési mechanizmustól. Ez azt jelenti, hogy ha úgy döntünk, hogy egy bizonyos reprezentációs módszert használunk a rendszerünk fejlesztésénél, azzal nagymértékben meghatározzuk a következtetésre használható algoritmusok körét is. A kiválasztott reprezentációs nyelvnek számos feltételnek meg kell felelnie, annak érdekében, hogy biztosítsa a hatékony felhasználását. Ezek a követelmények a következők:

- Legyen kifejező és tömör.
- Legyen független az általa reprezentálni kívánt tudástól
- Legyen az általa leírt információ felhasználható következtetésre
- Legyen egyértelmű

Ezeket a feltételeket a matematikai logika többnyire teljesíti. Segítségével az adott szakterület ismereteit mondatok segítségével tudjuk leírni. A logika segítségével ezután a már ismert adatokból (állításokból, axiómákból) újabb mondatokat tudunk előállítani. Ez a folyamat a következtetés. Ugyanakkor a logika eszköztára lehetővé teszi számunkra, hogy egyes állítások igazságtartalmát vagy akár több állítás ellentmondás-mentességét is vizsgáljuk.

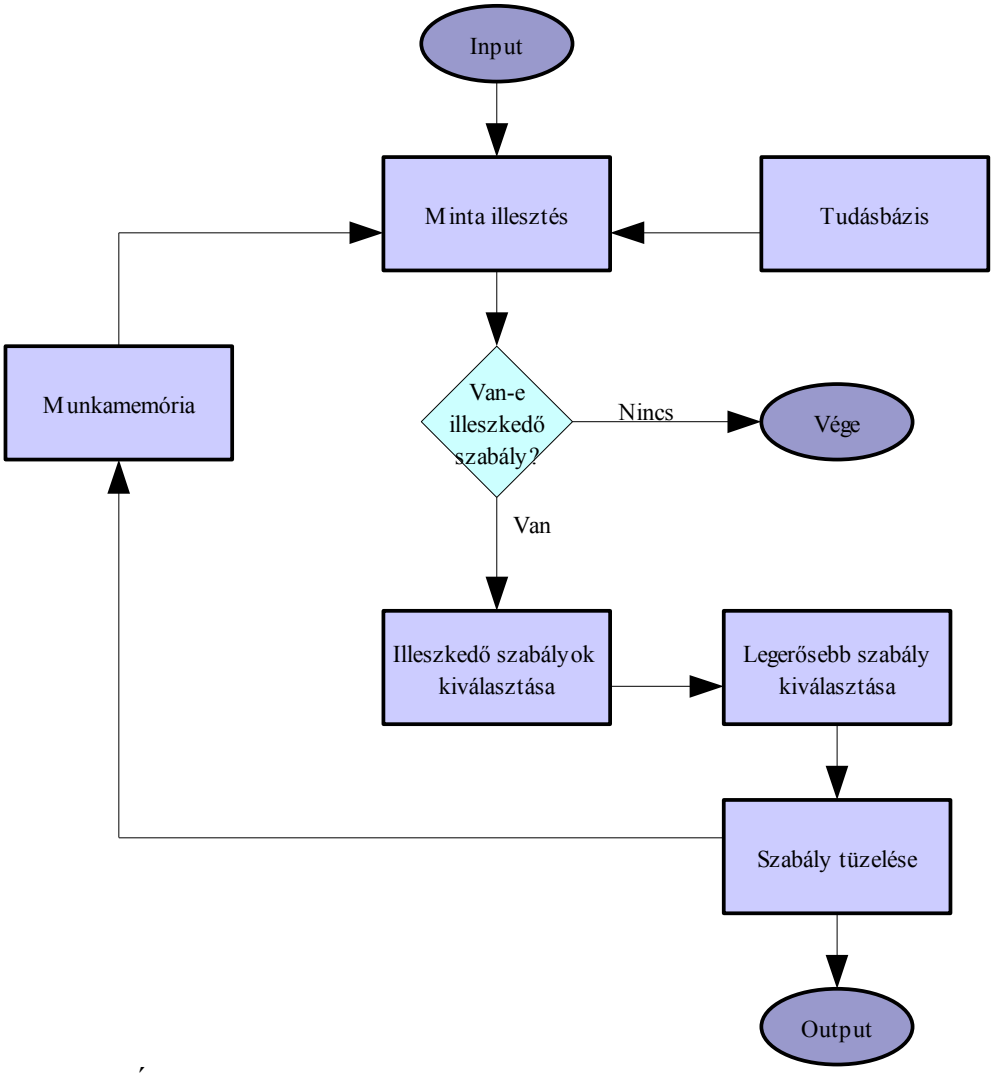
3.7.2 Szabályalapú tudásreprezentáció

A tudásalapú rendszerek fejlesztésében jelenleg legáltalánosabban alkalmazott szabályalapú következtetés is a logikán alapul – mégpedig az elsőrendű logikán, vagy más néven predikátumkalkuluson.

A szabályalapú rendszerek tudásbázisa két részből áll: az adatbázisból, és a szabálybázisból. Az adatbázis – melyet szokás munkamemóriaként is emlegetni – elemi tényállítások halmaza. Ez rögzíti mindazon tényeket, melyeket egy probléma megoldásának adott pillanatában a világról és magának a feladatnak a megoldásáról ismerünk. Minden állítás, ami nincs benne az adatbázisban, hamisnak tekintendő.

A tudásbázis másik fő része a szabálybázis, mely HA – AKKOR (IF – THEN) típusú szabályokból áll. Egy szabály egy feltétel – reakció páros, amit úgy kell értelmezni, hogy amennyiben a feltétel rész teljesül (igaznak értékeli ki a következtető rendszer), a reakció részt végre kell hajtani. A feltétel oldal kifejezi azokat a körülményeket, melyek között egy szabály egyáltalán aktivizálódhat. A feltételt egy minta írja le, amely adatbázisbeli tényekre vonatkozik. A mintában tények logikai kapcsolata definiálható (ÉS, VAGY), illetve tagadás fogalmazható meg. A minta tartalmazhat változókat is. Az akció oldal adja meg, hogy mi történjék a szabály aktivizálódása (ún. tüzelése) nyomán. Egy szabálynak kétféle hatása lehet: módosíthatja az adatbázist újabb elemi tényállítások hozzáadásával vagy korábbiak törlésével, illetve, amolyan mellékhatásként, input/output feladatokat láthat el.

Adatvezérelt következtetés



Ábra 4: Szabályalapú rendszerek működése (adatvezérelt)

A 3-as számú ábrán jól látható, hogyan működik a szabályalapú rendszer következtető motorja. Elsőként a tudásbázisban szereplő állításokkal kiértékeli a következtető motor a szabálybázisban található szabályokat. Ez a mintaillesztés. Amennyiben vannak illeszkedő szabályok, valamilyen mechanizmus alapján (pl. szabályok prioritásának figyelembevétele) kiválasztja, hogy melyik a legerősebb szabály, majd végrehajtja ennek a szabálynak a jobb oldalát. Ez a tüzelés. Ennek eredményeképpen módosulhat a munkamemória, illetve valamilyen output is megjelenhet. Ezután a folyamat megismétlődik, miközben jellemzően

bemeneti adatokat vár a következtető rendszer a felhasználótól. Ez egészen addig ismétlődik amíg lesz olyan szabály, amely tüzelhet – ha elfogyott az összes ilyen a következtető motor leáll. Ez az adatvezérelt következtetés, más néven „előre láncolás”.

Célvezérelt következtetés

Előfordulhatnak olyan esetek is, amikor nem az a célunk, hogy egy lehetséges választ, javaslatot adjunk a rendszerünkkel a felhasználónak hanem, hogy megtaláljuk annak a módját, hogyan lehet egy adott végkövetkeztetésre, eredményre jutni. Itt a szakértő rendszer által adott megoldás, tulajdonképpen az lesz, hogy a bemeneti adatokból és a tudásbázisban már szereplő kiinduló ismeretekből lehetséges-e egy adott eredmény levezetése. Ez a célvezérelt következtetés, vagy más néven „hátraláncolás”. Ennél a módszernél a következtető rendszer végignézi, hogy található-e olyan szabály a szabálybázisban, amelynek a THEN ágában szerepel a célként kitűzött érték. Amennyiben a rendelkezésre álló információk alapján az IF ágban szereplő feltételek kielégíthetőek, úgy megvan az eredmény: a cél elérhető, levezethető. Amennyiben nem tudjuk kiértékelni a kifejezést, vagy az eddigi ismeretek alapján hamisnak találja, a következtető rendszer célként felveszi a feltételben szereplő kifejezéseket és folytatja a keresést. A folyamat akkor fejeződik be, ha sikeresen megtaláltuk a levezetés módját, vagy ha nem találunk egyetlen olyan szabályt sem, amelynek THEN ágában szerepelne a cél.

A két következtetési módszer összehasonlítása

Összefoglalva, az adatvezérelt következtetés a tények ismeretében alkalmazza rájuk az összes lehetséges szabályt, ezzel újabb tényeket előállítva, míg a célvezérelt következtetés egy kitűzött célból indul ki és próbálja megtalálni azokat a tényeket, melyekkel elérhető az. Általánosságban elmondható, hogy mindkét módszer alkalmas a másik módszerrel megoldható problémák kezelésére, de ennek ellenére jelentős teljesítménybeli különbségek lehetnek közöttük, így egyes feladatok megoldásánál az egyik, míg másoknál a másik lehet a leghatékonyabb megoldás.

Például, ha arra vagyunk kíváncsiak, hogy egy betegnek mekkora esélye van arra, hogy a

tünetek alapján rákot diagnosztizáljunk nála, akkor a célvezérelt következtetés a legmegfelelőbb választás. Viszont, ha például arra szeretnénk választ kapni, hogy bizonyos tünetek mely betegség diagnosztizálását valószínűsíthetik, akkor az adatvezérelt következtetést célszerű választani. A döntésben egy ökölszabályt adhat nekünk, ha azt vizsgáljuk, hogy a tudásbeszerzés során előállított tudásbázisunkban milyen a szabályok és az adatok, tények viszonya. Amennyiben nagyon sok adatunk van, viszont aránylag kevés szabályunk úgy a célvezérelt következtetés lehet a legeredményesebb, mivel ott nem lesz szükség arra, hogy az összes adatra alkalmazzuk a szabályainkat. Amennyiben nagyon sok szabályunk van, de kevés kiinduló adatunk, úgy viszont az adatvezérelt következtetés adhatja a legjobb megoldást.

3.7.3 Esetalapú ismeretábrázolás és következtetés

A tudásreprezentáció egy másik népszerű formája az eset alapú ismeretábrázolás. Az ilyen rendszerek tudásbázisa sokkal inkább hasonlítható egy konkrét eseteket tároló lexikonhoz, semmint egy pontosan megfogalmazott szabályokkal teli halmazhoz. Nagy előnyének tekinthető, hogy az emberi gondolkozáshoz sokkal közelebb áll, mint a klasszikus szabály alapú reprezentáció. A tudásbázist korábbi esetek feldolgozásával, kielemzésével tölti fel a tudásmérnök. Minden egyes ilyen eset tartalmazza az adott probléma pontos elírását, a körülményeket, melyek a probléma keletkezésekor fennálltak és ami a legfontosabb, hogy tartalmazza ezen esetek megoldását, eredményét is. Maga az a folyamat, amelynek során, az esetet megoldó szakértő kikövetkeztette, hogy melyik megoldás vezet eredményre nincs letárolva. Ez ellentétben áll a szabályalapú ismeretábrázolással, ahol nyomon lehet követni, hogy hogyan jutott a szakértő (vagy az ő tudását magába sűrítő szakértő rendszer) a kapott eredményre.

Az esetalapú következtetésnél ahhoz, hogy megoldjunk egy esetet, hasonlítjuk azt az esetbázisban található korábbi esetekhez. Az eredményül kapott esetek felhasználásával a rendszer javaslatot tesz a megoldásra. Amennyiben a javaslattal megoldódik a probléma, ez a megoldott eset is felkerül az tudásbázisba, amennyiben szükség van rá, akár módosítva.

A következő lépések írják le, hogyan történik a hasonló esetek kiválogatása a tudásbázisból:

- Elsőként meg kell határozni, azokat a kulcsfontosságú tulajdonságokat, melyek a megoldásra váró problémát jellemzik.
- Az esetbázisból ki kell keresni a legjobban hasonlító eseteket, leggyakrabban valamilyen hasonlóságot kifejező érték kíséretében.
- Ki kell választani azt az esetet, amely a legnagyobb hasonlóságot mutatja a megoldandóval.

Az esetalapú ismeretábrázolás alapgondolata az, hogy a szakértők általában nem egyedi problémákkal kerülnek szembe munkájuk során – jellemzően inkább ugyanazon problémák különböző variációival. Sokkal egyszerűbb egy problémára megoldást találni, amennyiben rendelkezésünkre állnak korábbi nagy hasonlóságot mutató esetek, mint az egész folyamatot újra és újra a semmiről kezdeni. A tapasztalatok is inkább azt mutatják, hogy a szakértők a valós helyzetekben is saját korábbi tapasztalataikra alapoznak, melyeket hasonló esetek megoldásakor szereztek.

Egy adott probléma megoldásakor az eset alapú következtető (legyen az akár egy program, akár egy emberi szakértő) megkeresi azokat a korábban megoldott eseteket, melyek nagyban hasonlítanak az aktuálishoz. A következtető gép ezután átülteti a megtalált eset megoldását a vizsgált esetre, és ha szükséges módosításokat végez azon, annak függvényében, hogy milyen adatokban és milyen mértékben tér el a régi eset az újtól. Ezután, amennyiben a probléma sikeresen megoldódott, ezt a módosított esetet is letárolja az ismeretbázisában, annak érdekében, hogy azt a későbbiekben újrahasznosíthatja. Ez is nagy hasonlóságot mutat az emberi szakértők tanulásával.

Általában elmondható, hogy az esetalapú következtető rendszerek a következő nagyobb alkotórészekből épülnek fel:

- Adatbázis, mely a múltban megoldott eseteket tartalmazza.
- Néhány eljárás, melyek lehetővé teszik a régebbi esetek előhívását, az újak letárolását.

- Szabályok, melyek segítségével megállapítható az egyes esetek közötti hasonlóság mértéke.
- Olyan eljárás mely lehetővé teszi a korábbi esetek megoldásának átültetését az újakra.

Első lépésben a következtető rendszer megpróbálja „megérteni” a problémát: összegyűjti azokat a jellemzőket, melyek egyértelműen azonosíthatják a problémát és elkülöníthetik azt más problémaosztályoktól. Ezek a kulcs attribútumok indexként szolgálnak az adatbázisban: segítségükkel könnyen és gyorsan elő lehet keresni a hasonló eseteket. Ezután a hasonlító függvények megpróbálják megtalálni a leginkább hasonló esetet az adatbázisban. Az így megtalált esetek megoldásaira, mint lehetséges megoldásra tekint a rendszer. A legtöbb gyakorlatban használt tudásalapú rendszerben nagyon kicsi az esélye annak, hogy olyan esetet találjunk, amely teljes mértékben illeszkedik, ebből kifolyólag a rendszernek adaptálnia, azaz át kell dolgoznia a megoldást. Az adaptáló szabályok azt vizsgálják, hogy az egyek attribútumok melyekben eltérés mutatkozik, mennyiben változtatják meg a megoldási módszert. Ha sikeresen megoldódik a probléma, az eset eltárolásra kerül, annak megoldásával együtt – ez tulajdonképpen a tanulási folyamat. Egyes rendszerek a sikertelen megoldásokat is eltárolhatják.

Az esetalapú rendszerek előnyei

Számos előnye van az esetalapú következtetésnek. Az olyan szakterületeken ahol hiányoznak a pontosan megfogalmazható szabályok, elméletek és a problémamegoldás leginkább tapasztalatszerzéssel javítható, a szabályalapú rendszereket csak jóval kevésbé hatékonyan lehetne alkalmazni. Ugyanígy, ha a kiindulási információk és a megoldások közötti kapcsolatokat csak nehezen lehet HA – AKKOR formában megadni, illetve nagyon sok a kivételes eset az adott szakterületen, szintén az esetalapú ismeretábrázolást kell választanunk. Nagy előnye az ilyen következtetésnek, hogy ugyanolyan jól tudja kezelni a gyakori és a ritka, vagy kivételes eseteket.

Az esetalapú rendszerek hátrányai

Ugyanakkor hátrányai is vannak az esetalapú rendszereknek: ha a probléma, amit meg kell oldaniuk, egy teljesen új problématípushoz tartozik, a rendszer nem fogja tudni helyesen kezelni azt, illetve ha a problémához nem talál közeli esetet, a szolgáltatott megoldás helytelen is lehet. A rendszer esetbázisának felépítéséhez rengeteg korábbi esetet kell feldolgoznunk, és előfordulhat, hogy ezek nem állnak rendelkezésünkre, mert például a szakértő még nem találkozott minddel. Az is előfordulhat, hogy a szakterületen felmerülő problémák annyira változatosak, hogy nem érdemes esetekben gondolkozni. Ilyenkor sokkal kézenfekvőbbnek bizonyulhat a szabály alapú ismeretábrázolás és következtetés.

3.8 Keretrendszerek

A korai szakértő rendszerek egyik jellemző tulajdonsága volt, hogy nagyon nehezen lehetett őket továbbfejleszteni, illetve kifejlesztett következtető mechanizmusokat más szakterületeken is alkalmazni, mivel ezek a rendszerek csak a saját specifikus szakterületükön voltak alkalmazhatóak. Minden egyes új rendszerhez az egész fejlesztési folyamatot az elejéről kellett kezdeni. Ennek megoldására az 1980-as 90-es években elkezdtek keretrendszereket, ún. shelleket fejleszteni. Ezt úgy érték el, hogy kivonták a már kész rendszerekből a szakismeretet, és csak az üres vázat kínálták a felhasználóknak. Ezzel az általános célú eszközrendszerrel olyan fejlesztői környezetet biztosítottak a programozóknak, mely nagymértékben megkönnyítette az intelligens rendszerek fejlesztését. A ma működő tudásalapú rendszereket szinte kizárólag ilyen keretrendszerek segítségével készítik, melyek a következő funkciókkal rendelkeznek (rendelkezhetnek):

- Integrált szabály és keret alapú tudásreprezentáció és következtetés.
- Eset alapú reprezentáció és következtetés.
- Integrált fejlesztői környezet, mely egyszerre támogatja a tudásbeszerzést, lehetőséget ad magyarázó alrendszerek készítésére.
- Lehetővé teszik a tudásbázis karbantartását, fejlesztését.
- Grafikus felhasználói felület.

- Adatbázis interfész a legtöbb széles körben használt adatbázishoz.
- Használhatóak több operációs rendszer alatt.
- Lehetővé teszik más rendszerekkel való zökkenőmentes együttműködést, összetett, esetleg más rendszerekbe is beágyazott mesterséges intelligencia rendszerek fejlesztését.

3.9 Történeti áttekintés

A következőkben megpróbálok röviden bemutatni néhány fontosabb tudásalapú, illetve szakértő rendszert, melyeket szinte minden szakirodalomban megemlítenek.

3.9.1 DENDRAL (1965-1983)

A DENDRAL volt az egyik legelső szakértő rendszer. Kifejlesztői azt a célt tűzték ki, hogy egy olyan rendszert alkossanak, mely a tudományos ismereteket és következtetést egy speciális tudományterületen, a szerves kémiában képes alkalmazni. Másik elképzelésük az volt, hogy olyan MI technológiát hozzanak létre, mellyel képesek jobban megérteni a tudományterület alapvető problémáit, ami nagymértékben elősegítheti a rendszer által szolgáltatott megoldások magyarázatának lehetősége. Tudásbázisa segítségével képes volt eddig ismeretlen szerves anyagok felépítésének meghatározására és képességei sok esetben az emberi szakértőkővel vetekedett, így számos szakirodalomban jelentek meg a segítségével készített publikációk. A vegyészek, biológusok és informatikai szakemberek együttműködésével elkészített DENDRAL, nemcsak a szabályalapú szakértőrendszerek létjogosultságát és erejét szemléltette, hanem olyan gyakorlatban is sikeres alkalmazott rendszerré vált, melynek eszköztárát ugyanúgy használták az iparban, mint az egyetemeken.

3.9.2 MYCIN (1972-1980)

A MYCIN egy olyan interaktív rendszer, mely bizonyos fertőző betegségek diagnosztizálására használható. A rendszer javaslatot is tesz a kezelések módjára és részletes magyarázattal is szolgál arról, hogyan jutott az adott következtetésre. Szabályozott

körülmények között az eredményei elérték az orvos szakértőkét. Ugyanakkor a MYCIN számos eredményt produkált a mesterséges intelligencia területén is. Itt található meg elsőként a tudásbázis és a következtető mechanizmusok tudatos szétválasztása. Szabályalapú rendszere a célvezérelt következtetést (a hátraláncolás) használta. Mivel az orvoslás rendkívül gyorsan fejlődött, tudásbázisát úgy alkották meg, hogy könnyen bővíthető legyen újabb szabályokkal, ismeretekkel. Ugyanakkor mivel az orvosi diagnózisok egyik jellemzője, hogy szinte sohasem százszázalékosak, a MYCIN-be is beépítettek bizonytalanságot jelző faktorokat, melyekkel jelezni tudta egy adott következtetés megbízhatóságát, helyességének valószínűségét. Annak ellenére, hogy nem vált az orvosok mindennapi eszköztárának részévé, a MYCIN nagymértékben befolyásolta a későbbi MI kutatásokat. Olyan rendszerek születtek meg mintájára, mint: TEIRESIAS, EMYCIN, PUFF, CENTAUR, VM, GUIDON, és SACON rendszerek.

3.9.3 EMYCIN

A MYCIN következtető motorját használja, annak tudásbázisa nélkül, de ennek pótlására kibővítették egy a tudásbeszerzést megkönnyítő alrendszerrel is. Neve az „Essential MYCIN”-ből ered, ami arra utal, hogy csak a MYCIN leglényegesebb, szakterülettől független részeit kínálja a felhasználóknak, ezzel tulajdonképpen egy keretrendszert (shell-t) alkotva. Segítségével olyan szabályalapú szakértő rendszereket lehet készíteni, melyek tanácsokat adnak olyan területeken, melyek hasonlítanak az orvoslás és diagnóziskészítésben előforduló problémákhoz.

4. Tudásbeszerzés a PCPACK 5 rendszer segítségével

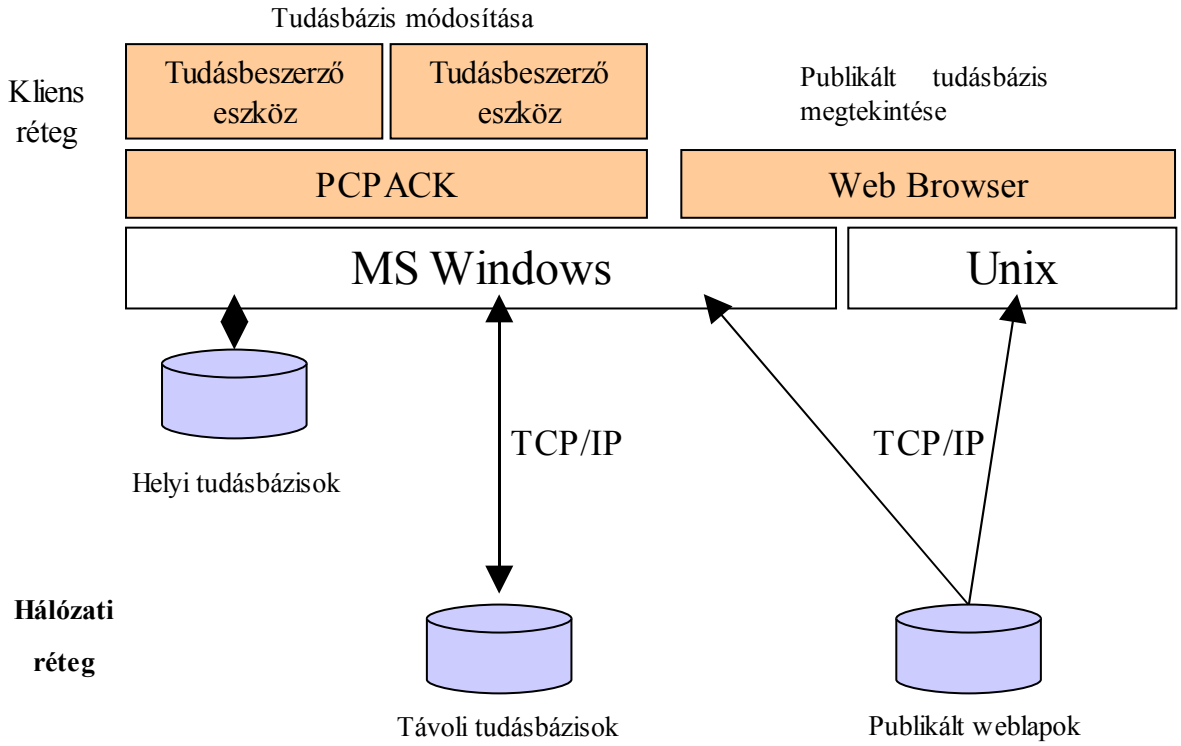
4.1 A rendszer bemutatása

4.1.1 Telepítés és rendszerkövetelmények

A rendszer kipróbálható változata a http://www.epistemics.co.uk/ oldalon regisztráció után érhető el. A regisztrációnál megadott email címre kapunk egy üzenetet, mely tartalmazza a telepítéshez szükséges kulcsot. A telepítéshez a következő feltételek szükségesek:

- *Operációs rendszer:* Windows XP Service Pack 1, Windows 2000 Service Pack 2, Windows NT Service Pack 5
- *Egyéb:*
  - Microsoft Data Access Components (legalább 2.8-as verzió)
  - Microsoft Jet 4.0 Service Pack 8
  - Microsoft Internet Explorer Version 5.5 (javasolt: 6-os verzió)
  - Adobe SGV Viewer plug in (az Internet Explorer-hez szükséges, hogy meg tudja jeleníteni a rendszerrel készített diagramokat).
  - Microsoft Word 97 (ahhoz szükséges, hogy Word dokumentumokból és HTML fájlokat is lehessen elemezni a rendszerrel)
- *Hardverkövetelmények:*
  - Hálózati kártya (amennyiben központosított tudásbázis szervert használunk)
  - Legalább 1Ghz-es processzorMinimum 512 Mb memória
  - 30 Mb szabad terület a program telepítéséhez (javasolt kb. 500 Mb szabad hely a tudásbázisokhoz)

4.1.2 A rendszer architektúrája



Ábra 5: A PCPACK rendszer architektúrája

Az 5-ös ábrán látható, hogyan épül fel a rendszer. A kliens rétegben található maga a PCPACK alkalmazás, melynek futtatása csak Microsoft Windows operációs rendszerek alatt lehetséges. Szintén kliens oldalon található a web böngésző is, amely már tulajdonképpen bármely platformon használható. Az operációs rendszer egy fájljaként jelenik meg a konkrét tudásbázis, mely lehet helyi, illetve távoli is. Ez utóbbi esetben az eléréshez hálózati kapcsolat szükséges és TCP/IP kommunikáció. Ugyanez elmondható, ha egy távoli gépem egy tudásbázisból generált weblapot akarunk megtekinteni a böngésző segítségével.

4.1.3 Eszközrendszer

A PCPACK olyan eszközrendszert ad a tudásmérnökök kezébe, melyek segítik a tudás megszerzését, megosztását, kezelését és újrafelhasználását. Ezek az eszközök többek között a következő tudásbeszerzési folyamatokat támogatják:

- Jegyzőkönyvelemzés
- Lépcsőző technikák
- Diagramok készítése (folyamatábra, állapot diagram, kapcsolatháló)
- Mátrix technikák

Ezek az eszközök a kliensrétegben dolgoznak, mindegyikük ugyanazon tudásbázist módosítja, kezeli (egy projekten belül). Így nem fordulhat elő inkonzisztencia a rendszerben. Mint korábban említettem, maga a tudásbázis fizikailag lehet az alkalmazást futtató számítógépen, vagy valahol a hálózaton egy dedikált tudásbázis szerveren. Ez utóbbi esetben lehetősége van a rendszer adminisztrátorának szabályozni a keretrendszer felhasználóinak hozzáférési jogait, mely remek eszköz arra, hogy elkülönítsük a tudásmérnököket az átlagfelhasználóktól.

A rendszer egyik nagy előnye, hogy rendkívül könnyen használható, grafikus felhasználói felületen keresztül teszi lehetővé a tudásbeszerzést. Bármilyen módosítást végzünk az egyik eszköz segítségével, az egyből megjelenik a tudásbázisban, így a rendszer többi része is azonnal az új állapotot fogja kezelni. Lehetőségünk van az elkészített tudásbázist weblap formájában publikálni, mely lehetővé teszi a tudásbázis hatékony felhasználását a gyakorlatban, hiszen bárki elérheti azt egy böngésző segítségével. Maga az így elkészített tudásbázis nem több, mint amit neve sugall – nincs beépített következtető mechanizmus a PCPACK-ban, csak az ismeretek beszerzését, rendszerezését és megjelenítését szolgálja. Ebből következik viszont egyik legnagyobb előnye: bármilyen rendszert is akarunk készíteni, nagy segítségünkre lehet – függetlenül a választott ismeretábrázolási, következtetési technikáktól.

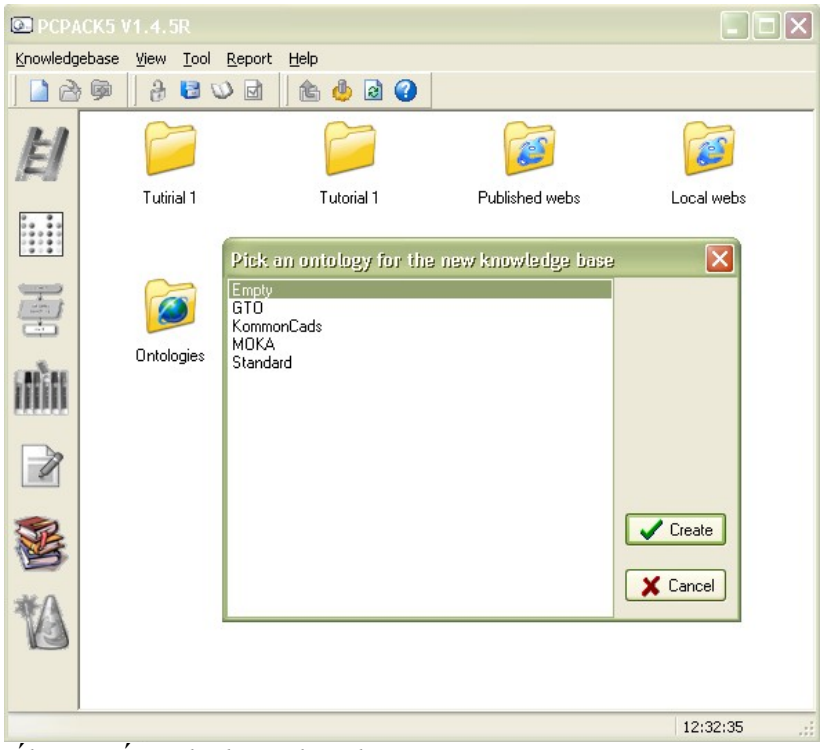
Korábban említettem, hogy a tudásalapú rendszerek készítésének egyik legfontosabb része az ismeretbeszerzés, mivel ha ezt nem jól végzi el a tudásmérnök, akkor a rendszer sikerére kicsi az esély. A PCPACK lehetőségeinek alapos tanulmányozása után úgy döntöttem, hogy ez lesz az az eszközrendszer, mellyel a saját szakértő rendszeremhez szükséges ismereteket beszerzem. Ez a folyamat valamelyest különbözik a korábban említettől és nem is lesz lehetőségem minden módszer használatára – ennek oka, hogy ebben az esetben én magam vagyok a tárgyköri szakértő és a tudásmérnök is. Ugyanakkor az ismeretek egy részét különböző szakirodalmakból nyertem, de gyakran fordultam az online közösségek szakmai fórumaihoz is.

Ebben a fejezetben csak magának a tudásbeszerzésnek a folyamatát elemzem, annak leprogramozásáról – és ezáltal egy működőképes rendszer létrehozásáról – a következő fejezetben lesz szó.

4.2 A rendszer használata

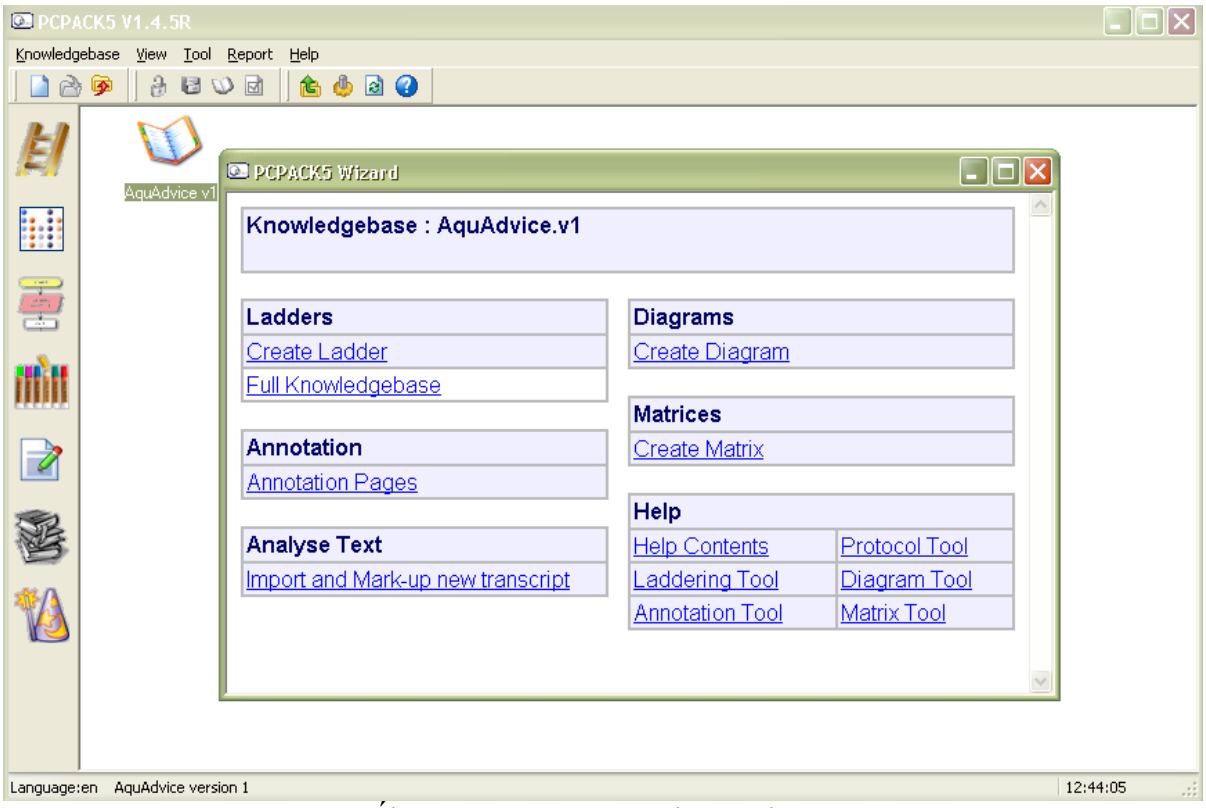
4.2.1 Új tudásbázis létrehozása

A PCPACK elindítása után a *Knowledgebase* → *New* menüpont alatt hozhatunk létre új tudásbázist. Itt lehetőségünk van előre elkészített ontológiák használatára, melyek célja, hogy kiindulási pontot adjanak az ismeretbázis elkészítésében (6-os ábra). Én egy teljesen üres tudásbázist hoztam létre, hogy az összes lépést végigkövethessem.



Ábra 6: Új tudásbázis létrehozása

Miután a *Create* gombra kattintunk, a rendszer egy párbeszéd ablakban várja, hogy megadjuk a leglényegesebb információkat a tudásbázisunkról (többek között: a nevét, verziószámát, a projekt nevét, amihez létrehozzuk és a különböző kapcsolattartó embereket). A *Save* gombra kattintva a rendszer létrehozza a szükséges fájlokat és elkezdhetünk dolgozni. A tudásbázisunk megnyitása után a 7-es ábrán látható ablak fogad minket. Itt jelennek meg különböző kategóriákba csoportosítva, azok az eszközök, melyeket a PCPACK nyújt a tudásbeszerzéshez.



Ábra 7: A PCPACK eszközrendszere

4.2.2 Jegyzőkönyv elemzési technika

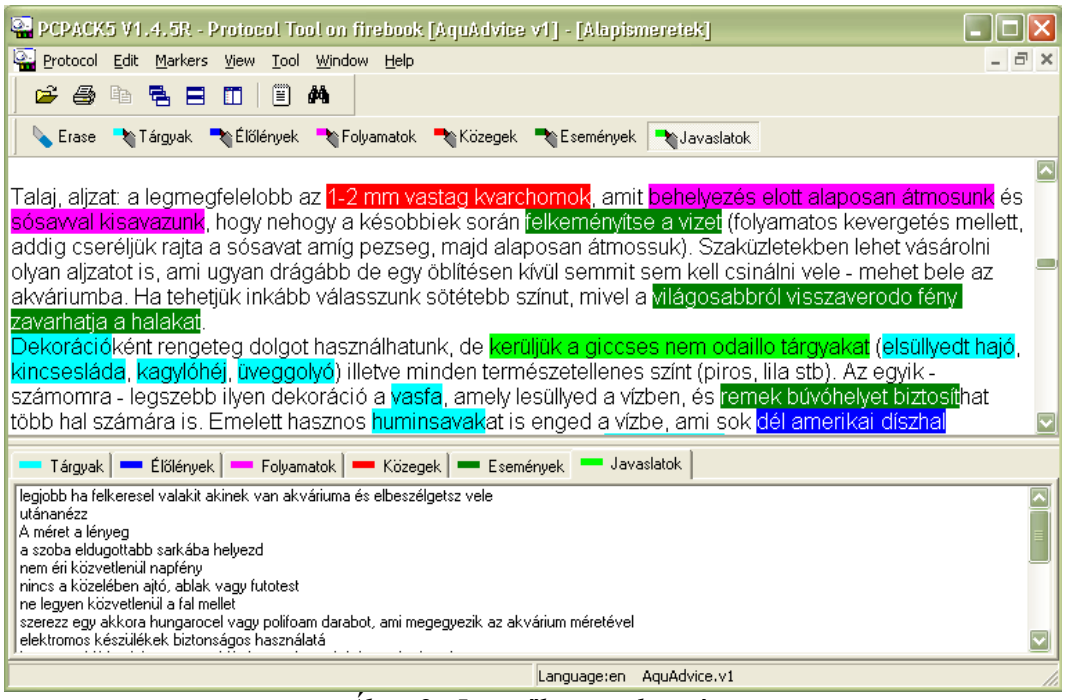
Első lépésként egy általam írd dokumentum elemzését választottam, annak érdekében, hogy az akvarisztikában használatos alapfogalmakat és lényeges összefüggéseket megtaláljam. Az elemzés alapjául szolgáló írást magam készítettem azzal a céllal, hogy a kezdő akvaristáknak nyújtson segítséget. Az írásban az alapokat ismertettem, úgy, hogy bárki – még ha sohasem hallott erről a hobbiról – megérthesse, miként kell sikeresen űzni azt. Ebből kifolyólag úgy véltem, hogy tökéletes kiindulási pontot nyújthat ez a forrás a szakértő rendszerem tudásbázisának elkészítéséhez.

A jegyzőkönyvelemzéshez az *Analyse Text* csoportba tartozó *Import and Mark-up new transcript* linkre kell kattintanunk, majd miután megadtuk a feldolgozandó fájl helyét, illetve a jegyzőkönyv nevét (*Alapismeretek*), megnyílik az eszközhöz tartozó ablak és megkezdhetjük a forrásszöveg feldolgozását. A PCPACK ezt úgy teszi lehetővé, hogy a

dokumentum megnyitása után különböző színekkel emelhetjük ki a szövegben felbukkanó hasznosnak ítélt információkat, ugyanúgy mintha papíron dolgoznánk szövegkiemelő tollal. Minden egyes ilyen kiemelés színe attól függ, hogy melyik nagyobb csoporthoz akarjuk kapcsolni az adott fogalmat. Ehhez viszont először szükséges meghatározni, azokat a szempontokat, amelyek szerint csoportosítani kívánjuk a szövegben előforduló kulcs adatokat. Én ezeket a következőképp definiáltam:

- *Tárgyak*: minden olyan eszköz beletartozik, amely bármilyen módon kapcsolatba hozható az akvarisztikával. Függetlenül attól, hogy az akváriumban vagy azon kívül használjuk
- *Élőlények*: az akváriumban élő halak, rákok és növények.
- *Tevékenység*: a hobbi űzése során felbukkanó rendszeres vagy akár egyszeri folyamat, melyek az akvarista aktív közreműködését igénylik.
- *Közeg*: minden, amivel a halak kapcsolatba kerülnek, például a víz, a talaj.
- *Közeg tulajdonsága:* a közeg azon jellemzői, amelyek fontosak az akvarisztikában.
- *Események*: a folyamatokkal ellentétben itt nem aktívan vesz részt az akvarista, többnyire csak felfedezi az esemény eredményét, megtörténtét. Leginkább az akváriumban lejátszódó folyamatoknak vagy élőlényeknek köszönhető.
- *Javaslatok*: tanácsok, ötletek melyeket érdemes megfogadni.

A későbbiekben természetesen ezek a csoportok tovább lesznek osztva, hogy részletesebben lehessen az ismereteket csoportosítani. A megnyitott ablakban (*Protocol Tool*) induláskor csak egy törlést szolgáló eszköz áll rendelkezésünkre. Különböző színek felvételéhez a *Markers* → *New* menüt kellett használnom. A fenti csoportoknak megfelelően hat új kiemelő színt hoztam létre.

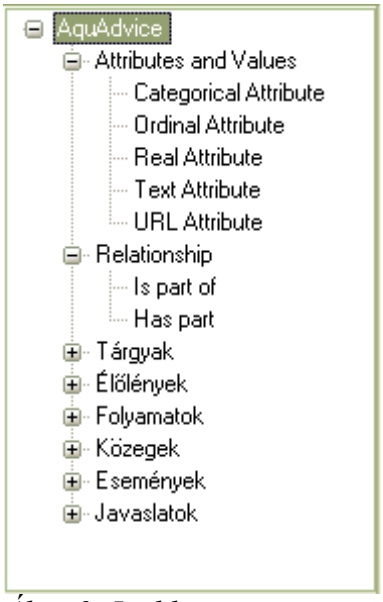


Ábra 8: Jegyzőkönyv elemzés

A szöveg alapos végigolvasása közben a megfelelő színű kiemelőket kiválasztva kijelöltem a hozzá tartozó szöveges információkat. Az elemzés egy közbenső fázisát szemlélteti a 8-as ábra. Miközben bejelöljük a fontosnak tartott információkat, a rendszer a háttérben folyamatosan dolgozik a tudásbázissal, így a változtatások azonnal kiírásra kerülnek. Az azonos színnel kiemelt fogalmakat a rendszer automatikusan egy csoportba rendeli. Miután végeztünk az elemzéssel, bezárhatjuk az eszköz ablakát. Majd a varázsló segítségével hozzáláthatunk a folytatáshoz (7-es ábra).

4.2.3 Ladder készítése (lépcsőző technika)

A következő eszköznek ezt célszerű választani, hogy a jegyzőkönyvelemzéssel eddig összegyűjtött információkat rendszerezzük, jobban átláthatóvá tegyük és az esetleges hibákat, hiányosságokat javítsuk. Az eszközt a *Ladders* csoportban a *Create Ladder* linkkel indíthatjuk el. Miután itt is megadtunk egy nevet (*Alapismeretek*), egy üres ablak fogad minket. A 9-es ábrán láthatóak azok a csoportok, melyeket korábban hoztam létre, mielőtt nekifogtam volna a forrásszöveg elemzésének.



Ábra 9: Laddering

Itt ezen csoportokon kívül még kettőt láthatunk, melyek alapértelmezésként megjelennek minden tudásbázisban. Az *Attributes and Values* csoporton belül öt elemet láthatunk. Ezeket arra használjuk, hogy a tudásbázisban szereplő más objektumokhoz tulajdonságokat és az azokat kifejező értékeket rendelhessünk. Például az *akvárium* objektumnak, lehet *térfogat, magasság* stb. tulajdonsága (*Attributes*), és ezekhez rendelhetünk konkrét értékeket is, például 100 liter, 70cm stb (*Values*). A tudásbeszerzés kezdeti szakaszában, ezekkel még nem foglalkoztam, elsődleges célom az volt, hogy kialakuljon egy olyan hierarchikus rendszer, amelyben minden a szakterületen fontos objektum fellelhető legyen, és jól láthatóvá váljanak a köztük lévő kapcsolatok (*Relationship*). Ehhez viszont meg kellett határoznom, hogy a két beépített kapcsolaton kívül (*Is part of, Has part*), milyen más viszonyok lehetségesek az objektumok között (bár itt nem látható a rendszer ismeri még az „*Is a*” relációt is – ez az alapértelmezés minden objektum között).

A következő táblázatban láthatóak az általam használtak (***A*** betűvel hivatkozom a relációk bal oldalán, ***B***-vel pedig a jobb oldalán szereplő objektumokat):

Is part of	***A*** része ***B***-nek
Has part	***A***-nak részét képezi ***B***
Is a	***A*** a ***B*** konkretizálása
Szükséges hozzá	***A***-hoz szükséges ***B***
Részfolyamata	***A***-nak részfolyama ***B***
Előfeltétele	***A*** feladat elvégzése megköveteli ***B*** elvégzését

Táblázat 2: Kapcsolatok a tudásbázis objektumai között

Új kapcsolatok létrehozására a *Laddering → Properties → Create Relation* menüpont alatt lehetséges. Itt meg kell adnunk az új reláció nevét, valamint lehetőségünk van az inverz reláció meghatározására is (a PCPACK-ban a „Részfolyamata" relációt, a könnyebb olvashatóság és egyértelműség miatt a következőképpen neveztem el: *„-nak részfolyamata a"*). Kiválaszthatjuk azt is, hogy milyen típusú objektumok között létesíthető ez a fajta kapcsolat.

A következő lépésben minden egyes nagyobb csoportot egyenként kezdtem feldolgozni, úgy hogy kisebb alcsoportokat hoztam létre bennük, majd azokba rendszereztem a jegyzőkönyv elemzés során összegyűjtött fogalmakat. Ezt rendkívül kényelmesen *Drag and Drop* módszerrel teszi lehetővé a PCPACK. Megjegyzem, hogy ekkor végeztem el a jegyzőkönyv elemzés során felvett objektumok megnevezésének is pontosítását (átnevezését), annak érdekében, hogy egységesebb képet mutassanak, és következetesek legyenek. Számos olyan fogalommal találkoztam, melyek egynél több alkategóriába is besorolhatóak (például a Tubifex mint táplálék, lehet élő, fagyasztott, vagy szárított is), ennek kezelésére a PCPACK lehetővé teszi, hogy egy objektum (csomópont – *Node*) akár több szülővel is rendelkezzen, ezzel tulajdonképpen egy gráf szerkezetet létrehozva. Megjelenítéskor választhatunk, hogy fa struktúrát, vagy gráfot szeretnénk-e látni. Az előbbi esetben az adott objektumot kétszer (vagy többször) jeleníti meg – mindegyik szülő gyermekeként (az esetleges változtatások, például

átnevezés természetesen mindegyik „előfordulást” módosítani fogja). Amikor két objektum között kapcsolatot határozunk meg, azt a rendszer automatikusan *Is a* relációként értelmezi. A kapcsolatot szimbolizáló egyenesen jobb egérkattintással behozható egy menü melyben a *Change relation type* menüpontot kiválasztva lehetőségünk van az alapértelmezett kapcsolat megváltoztatására. A könnyebb áttekinthetőség érdekében, – miután minden egyes objektum között kiválasztottam a megfelelő típust – ugyanitt a *Relation type style* menüpont alatt beállítottam az egyes kapcsoltakhoz az őket szimbolizáló egyenesek megjelenését (szín, szaggatott vonal, nyílra végződés, vastagság stb.). Elsőként az *Is a* kapcsolatok meghatározását végeztem el, így az egyes alap csoportok (élőlények, közegek, események, stb.) között éles határok alakultak ki. Ezzel az volt a célom, hogy a lehető leghamarabb sikerüljön egy olyan struktúrát kialakítani, mely a későbbiek folyamán könnyen bővíthető. A kapcsolatok meghatározásának második szakaszában, már az egyes csoportok objektumai más csoportokban szereplőkkel is összeköttetésbe kerültek a többi reláció segítségével. Így eredményül egy rendkívül összetett irányított gráfot kaptam, melyben minden él a relációk egyikével van felcímkézve. Már ebben a fázisban is fedeztem fel olyan összefüggéseket, melyekre nem is gondoltam korábban, tehát látható, hogy a lépcsőző technika valamelyest elősegíti a passzív tudás megragadását. Az *8.1-es számú függelékben* látható a lépcsőző technika egyik közbenső lépésének eredménye.

4.2.3 Eldobható prototípus

A következő lépésben újra átvizsgáltam az összes eddig megszerzett ismeretet és annak érdekében, hogy ellentmondásmentessé tegyem a tudásbázisomat, egy teljesen új tudásbázist hoztam létre az előzőleg elkészített alapján. Itt már tisztában voltam a rendszer nyújtotta lehetőségekkel és korlátokkal, illetve kialakult bennem egy sokkal strukturáltabb kép a szakterületről. Így az előzőleg elkészített tudásbázis egy eldobható prototípusnak tekinthető, melynek segítségével a szakértő (jelenesetben én magam) újra átgondolhatja a szakterületén szerzett ismereteit és az azok között rejlő összefüggéseket. Mivel a rendszer folyamatirányítási feladatokat fog ellátni, ezért az új ismeretbázis felépítésének első szakaszában, a rendszer magját képező ismereteket – magukat a tevékenységeket – vittem fel a PCPACK-ba.

Ezután minden egyes folyamatot egyenként vizsgáltam, és meghatároztam azokat az objektumokat, amelyek valamilyen kapcsolatban lehetnek vele. Így folyamatosan bekerültek az adatbázisba a különböző eszközök, élőlények, stb. Eredményül elértem, hogy bármely objektum, amely szerepel a rendszerben, valamilyen módon részt vesz a szakterület folyamataiban, vagy azok egy részében. Ezzel elkerültem olyan ismeretek tárolását, melyek nem kerülnek felhasználásra, így jelentősen lecsökkentettem a tudásbázis méretét. A végleges osztályok a következők lettek:

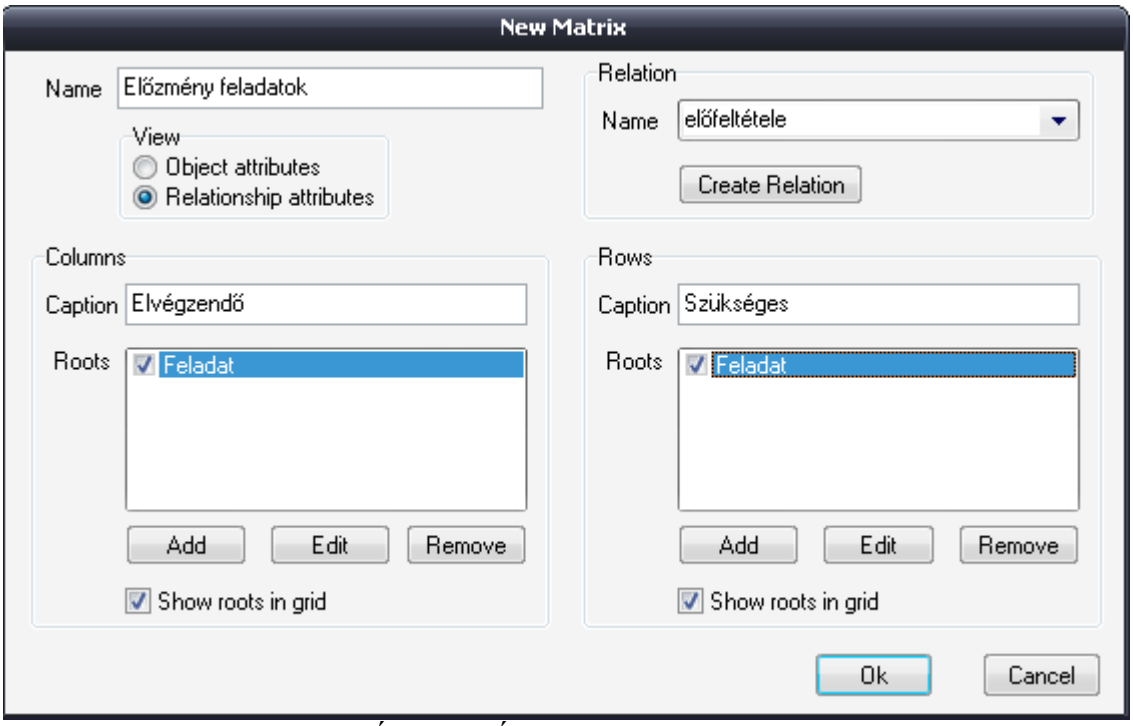
- *Akvarizálás:* Az összes elképzelhető folyamat, tevékenység, amellyel az akvarista találkozhat.
- *Élőlények:* Az akváriumban élő növények és állatok.
- *Tárgyak:* Minden olyan eszköz, amit valamilyen módon használ az akvarista.
- *Közeg:* Az akvárium élőlényei ebben élnek.
- *Feladatok:* Az akvarizáláshoz szükséges elvégzendő feladatok. Előfordulhat, hogy egy feladatnak vannak részfeladatai, vagy saját maga is egy másik részfeladata.

A PCPACK lehetőséget ad arra, hogy az egyes elemekhez tulajdonságokat (*Attribute*) rendeljünk. Korábban már láttuk, hogy alapértelmezésként a rendszer öt típus osztállyal rendelkezik. Ha egy új tulajdonságot rendelünk valamely objektumhoz, ki kell választanunk, hogy ezek közül melybe tartozik. Az öt kategória a következő:

- *Categorical*: akkor használjuk, ha a tulajdonság az értékeit egy megadott szimbólum tartományból veheti fel. Például a „*Feladatok*” típusú objektumokhoz hozzárendeltem egy *Gyakoriság* tulajdonságot. Ez négy értéket vehet fel (*Values*): *Naponta, Hetente, Havonta, Alkalmi.*
- *Ordinal*: Egész értékekkel kifejezhető tulajdonságok, például az akvárium űrtartalma, vagy egy feladat elvégzéséhez szükséges percek száma.
- *Real*: Valós értékű tulajdonságoknál használatos, az AquAdvice rendszer elkészítése során nem volt rá szükségem.
- *Text*: Rövid szöveges leírást tesz lehetővé.
- *URL*: Hosszabb leírások tárolása, melyben internetes hivatkozások is elhelyezhetőek.

4.2.4 A kapcsolat mátrix

A PCPACK lehetővé teszi, hogy a tudásbázisban tárolt adatok közötti kapcsolatokat egy mátrix segítségével határozzuk meg. Az eszköz nagy előnye abban rejlik, hogy segítségével, sokkal hatékonyabban tudunk olyan objektumok kapcsolatait letárolni, melyek egyszerre sok másik objektummal vannak relációban. Az eszköz használata úgy történik, hogy miután kiválasztottuk a reláció típusát, egy kétdimenziós táblázatban megjelöljük azon cellákat, amelyekhez tartozó főelemeket összekapcsolja az adott reláció. Az eszközt a PCPACK főablakából a *Tool → Matrix* menüpont alatt érhetjük el. A következő ablakban ki kell választanunk, hogy mely relációt szeretnénk használni a mátrixban, illetve melyek lesznek az oszlopokban, illetve a sorokban szereplő objektumok.



Ábra 10: Új mátrix létrehozása

Az AquAdvice rendszerben az „előfeltétele” reláció az egyik legösszetettebb. Az akvarizálás során felmerülő tevékenységek nagyobb része megköveteli, hogy bizonyos feladatokat már elvégezzünk előtte. Például, nem lehet úgy megetetni a halakat, hogy nem üzemeltük be az akváriumot vagy nem vettünk halakat. Ezen összefüggések megadására tökéletesen alkalmas a mátrix. A mátrixban használt relációnak ezt választottam, míg az oszlopok és sorok is

feladatokat tartalmaznak (10-es ábra). Az eredményül kapott mátrix egy részlete látható a 11-es ábrán.

Szükséges \ Elvégzendő	Akvárium előkészítése	Víz előkészítése	Víz beöntése	Akvárium vízzel feltöltése	Eszközök elhelyezése	Eszközök megfelelő helyének kiv	Eszközök bekapcsolása	Technika beüzemelése	Akvárium üzembehelyezése	Szükséges eszközök meghatáro:	Növények kiválasztása	Tartani kívánt halak kiválasztása	Akvárium megtervezése	Akvárium beüzemelése	Táptabletták elhelyezése	Tápoldat beöntése	Növények táplálása	Vas szint ellenőrzése
Akvárium előkészítése				□				□										
Víz előkészítése			□															
Víz beöntése																		
Akvárium vízzel feltöltése								□										
Eszközök elhelyezése							□											
Eszközök megfelelő helyének kivál					□													
Eszközök bekapcsolása																		
Technika beüzemelése																		
Akvárium üzembehelyezése																		
Szükséges eszközök meghatároz:																		
Növények kiválasztása										□								
Tartani kívánt halak kiválasztása										□	□							
Akvárium megtervezése									□					□				

Ábra 11: A kapcsolat mátrix egy részlete

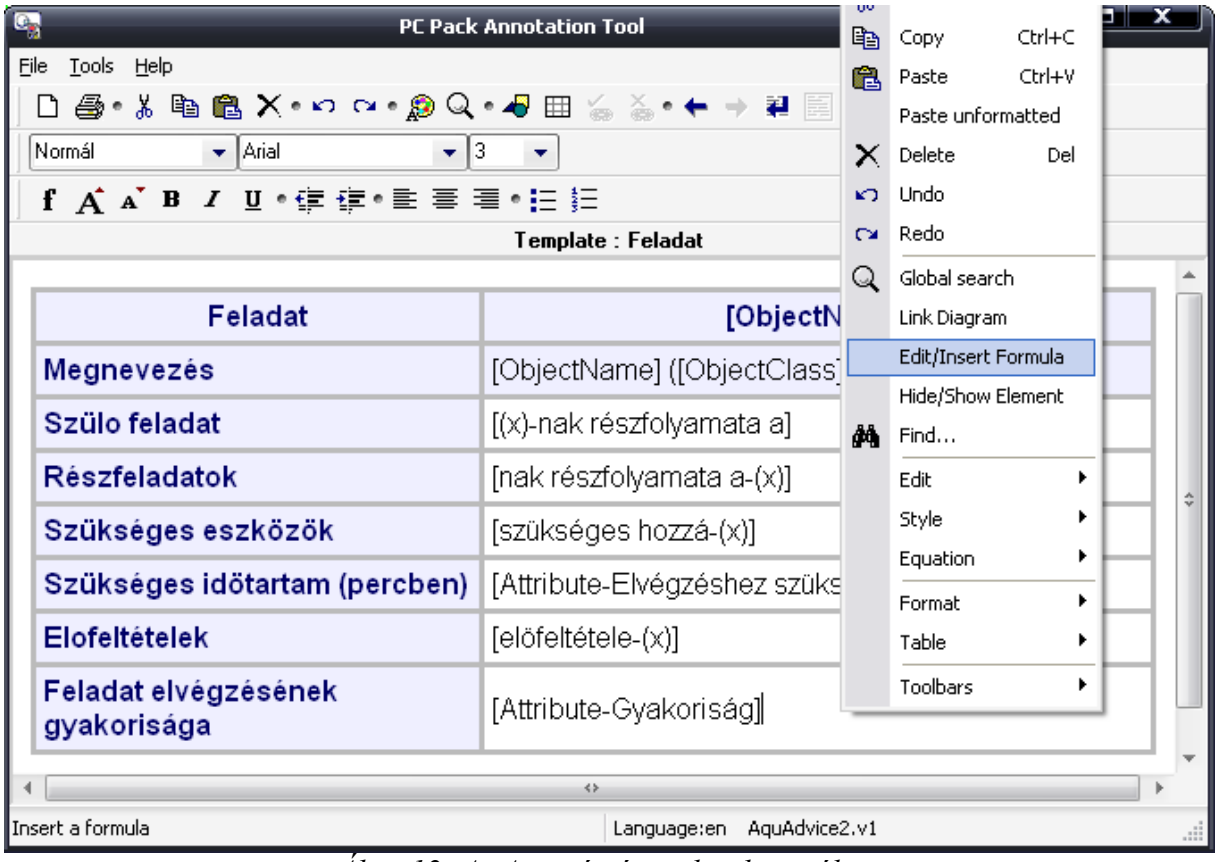
4.2.5 Az annotációs eszköz

Miután végeztem a csoportosítással és a kapcsolatok meghatározásával szükségem volt arra, hogy az eddig letárolt ismereteket újból ellenőrizzem, megjelenítsem. Kisebb tudásbázis esetén erre a célra tökéletesen megfelel a „laddering” során előállított fa- illetve gráfszerkezet, viszont egy nagyobb esetén szinte használhatatlan. A PCPACK egy úgynevezett Annotációs Eszköz *(Annotation Tool)* segítségével lehetővé teszi, hogy az egyes csoportokba tartozó objektumokat a hozzájuk tartozó információkkal együtt, más-más módon jelenítsük meg (12-

es ábra). Ehhez először el kell készítenünk egy-egy sémát (*Template*) az egyes csoportokhoz, melyben megadjuk, hogy milyen adatokat szeretnénk megjeleníteni. Eredményül egy könnyen átlátható táblázatot kapunk, melyben minden információ helyet kap, mely valamilyen módon köthető az adott elemhez. A tudásbázis áttekintését nagymértékben segíti, hogy minden objektumból elérhető a sémában megadott relációkkal hozzá kapcsolódó minden más objektum. Az eszköz bármelyik objektumon elérhető (*Jobbkattintás → Annotation Template)*, ekkor viszont figyelembe kell venni, hogy csak lefelé öröklődik a séma, a szülőkre nem lesz érvényes. Én minden egyes nagyobb csoportnak külön sémát hoztam létre.

Létrehozáskor meg kell adnunk, hogy milyen információkat akarunk látni az adott objektumról. A tevékenységek megjelenítésénél a következőket választottam ki:

- *Megnevezés:* az adott tevékenység megnevezése
- *Szülő feladat:* annak a feladatnak a neve, amelynek az aktuális a részét képezi
- *Részfeladatok:* az aktuális tevékenység részfolyamatai
- *Szükséges eszközök:* azok az eszközök, amelyek szükségesek a feladat elvégzéséhez
- *Szükséges időtartam:* az adott feladat elvégzéséhez szükséges idő, percben kifejezve. Egyes feladatoknál ez az érték 0, ami azt fejezi ki, hogy nem meghatározható az elvégzéshez szükséges idő.
- *Elvégzés gyakorisága:* milyen gyakran kell elvégezni az adott feladatot.
- *Előfeltételek:* az adott feladat elvégzéséhez először el kell végeznünk az itt felsorolt feladatokat.



Ábra 12: Az Annotációs eszköz használata

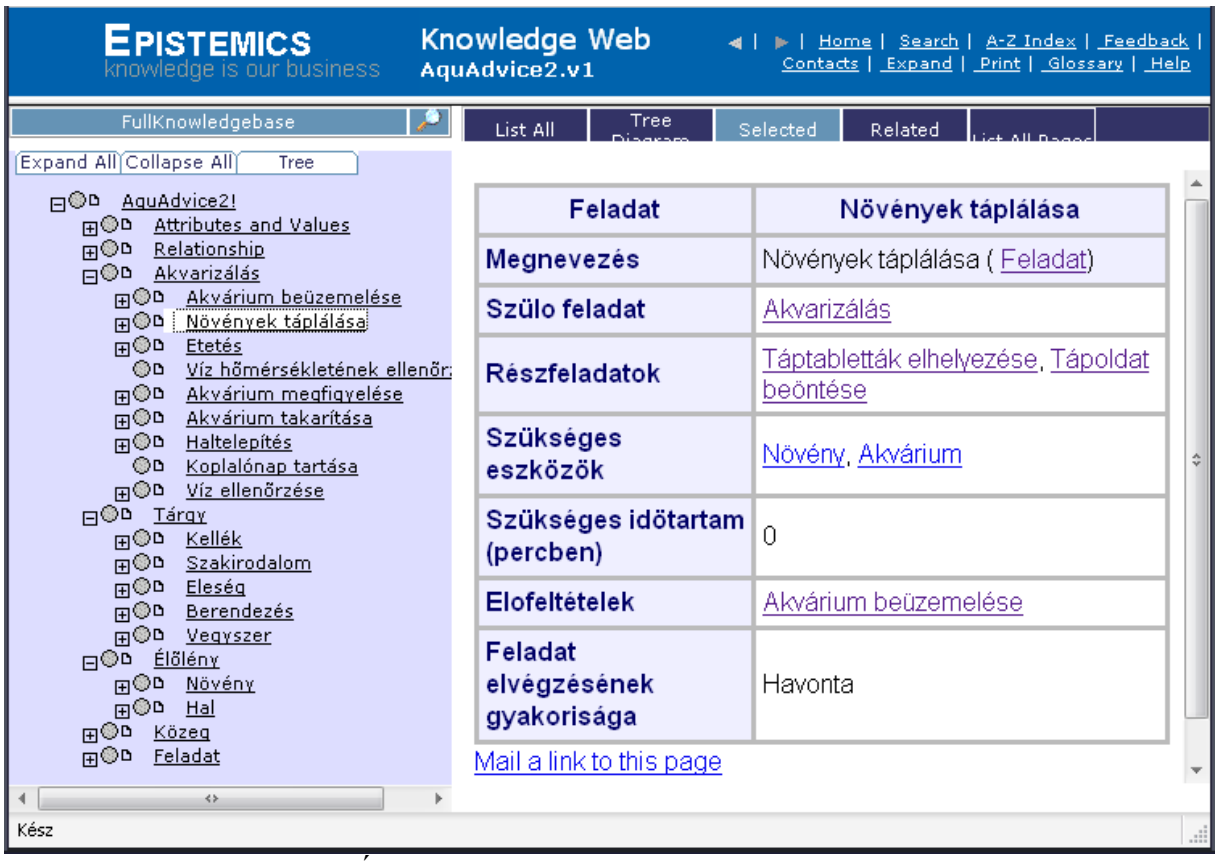
A séma megadásánál, természetesen nem explicit kell megadnunk az egyes mezők értékét. A rendszer ezeket automatikusan fogja generálni, a lépcsőző technikával előállított objektumstruktúra alapján. Ahhoz, hogy megadjuk, mely információk kerüljenek az adott mezőbe a 12-es ábrán látható módon az *Insert Formula* parancsot menüpontot kell használnunk *(Jobb klikk → Edit/Insert Formula)*. Itt lehetőségünk van a következők közül választani:

- Egy olyan reláció bal oldalán szereplő objektum, melynek jobb oldalán az aktuális objektum szerepel.
- Egy olyan reláció jobb oldalán szereplő objektum, melynek bal oldalán az aktuális objektum szerepel.
- Az adott objektum egy attribútuma
- Az adott objektum szülő osztálya, neve, stb.

Természetesen tetszőleges mennyiséget választhatunk a fentiek közül. Ennél az eszköznél lehetőségünk van még különböző szöveg, illetve táblázat formázási műveletek elvégzésére is.

4.2.6 A tudásbázis publikálása

Ezzel elkészült az a tudásbázis, amelynek segítségével hozzá lehet fogni a rendszert felépítő szabályok implementálásához. A PCPACK lehetőséget ad az ismeretek HTML dokumentumként való megjelenítésére, így a könnyebb kezelhetőség érdekében ezt ki is használtam. Az exportáláshoz a PCPACK főablakában először is be kell zárnunk az éppen megnyitott tudásbázist. *(Jobb klikk → Close),* majd el kell indítanunk a publikálást végző alrendszert *(Tool → Publisher).* A megnyíló ablakban lehetőségünk van az exportálás formátumának kiválasztására – a rendszerben több előre elkészített HTML sémát is integráltak, de saját magunk is készíthetünk egyet.



Ábra 13: A webes publikáció eredménye

A publikálást a *Knowledgebase →Publish as...* menüpont alatt érhetjük el, miután kiválasztottuk a tudásbázisunkat. Az eredményül kapott HTML dokumentumok megtekinthetjük a *View → Prewiev web* menüpont alatt. (JavaScript engedélyezése szükséges). A 13-as ábrán az eredményül kapott oldal egy részlete látható.

5. Az AquAdvice rendszer implementálása CLIPS nyelven

5.1 A CLIPS rövid bemutatása

A CLIPS (http://www.ghg.net/clips/CLIPS.html) egy szakértő rendszerek fejlesztésére a NASA által megalkotott eszköz. 1986-os megjelenése óta számos fejlesztésen és változtatáson átesett. Sok más szakértő rendszer létrehozásához kifejlesztett nyelvhez hasonlóan a CLIPS is LISP szerű szintaktikával rendelkezik. A CLIPS háromféleképpen teszi lehetővé az ismeretek ábrázolását, melyek a következőek:

- Szabályok (*rules*) melyekkel elsősorban tapasztalatokon alapuló heurisztikus ismereteket ábrázolhatunk.
- Függvények (*deffunctions*, *generic functions*), melyekkel a procedurális ismereteket ragadhatunk meg.
- Objektumok, melyek az objektum orientált paradigma következő fogalmait használják: osztályok, üzenetkezelés, absztrakció, beágyazás, öröklődés és polimorfizmus. A szabályok illeszkedhetnek tényekre és objektumokra is.

5.2 A szakértő rendszer működésének áttekintése

A rendszer az akvarisztikában előforduló folyamatok elvégzésében segít a felhasználónak, aki a rendszer kérdéseire válaszolva lép azzal interakcióba. Elsőként a rendszer felsorolja azokat a főbb feladatokat, amelyek vezérlésére fel van készítve (itt nem jelennek meg az egyes főbb folyamatok részfolyamatai), majd a felhasználóra vár, akinek ki kell választania, melyik feladatot szeretné elvégezni. Amint kiválasztja a kívánt feladatot, a rendszer következtető motorja működésbe lép és meghatározza:

- azokat a részfeladatokat, amelyeket el kell végeznie a felhasználónak,
- azokat az eszközöket, amelyekre szüksége lesz,

- azokat a folyamatokat, amelyeket már biztosan elvégzett (mivel vannak olyan folyamatok, amelyeket csak bizonyos más feladatok elvégzése után lehet végrehajtani) és ezzel együtt
- azokat az eszközöket, amelyek már birtokában kell, hogy legyenek.

Ezután megjeleníti a szükséges információkat és újból a felhasználóra vár, hogy megadja, mely feladattal, akarja folytatni. Természetesen minden választásnál a rendszer felkínálja a lehetséges válaszokat a felhasználó számára.

5.3 A tények ábrázolása CLIPS-ben

A PCPACK segítségével előállított tudásbázist a publikáció által előállított formában használtam fel. Minden egyes objektumot egyenként vizsgáltam meg és előállítottam az adott ismeretrészletet reprezentáló szabályokat, illetve tényeket. Először a tények meghatározását végeztem el, melyeket két kategóriába osztottam: folyamatok és eszközök. Az utóbbi csoportba került a szakterületen megtalálható minden kézzelfogható dolog (halak, növények, eszközök stb.).

A folyamatokról a következő információkat tároltam le:

- *Megnevezés*: a folyamat neve.
- *Gyakoriság*: az elvégzésének gyakorisága.
- *Elvégzendő*: értéke *TRUE* vagy *FALSE* lehet *(T/F), TRUE* értékű abban az esetben, ha az adott feladatot el kell végezni (ezeket listázza ki a rendszer a felhasználónak, aki ezek közül választ).
- *Folyamatban*: *(T/F), TRUE* értéket az a feladat kap, amely a felhasználó által kerül kiválasztásra.
- *Elvégezve*: *(T/F), TRUE* értéket kap ha egy folyamat teljesen befejezettnek tekinthető, azaz minden részfeladatát elvégeztük.

A fentiek mellett még egyéb technikai jellegű információkat is tárolok az egyes feladatokról. A másik csoportba az eszközök tartoznak, melyekről a következő adatokat tároltam le:

- *Megnevezés*: az eszköz neve.
- *Kategória*: mely eszköztípusba tartozik.
- *Szükséges*: *(T/F)*, értéke *TRUE,* ha az adott eszköz szükséges a jelenleg folyamatban lévő feladatok elvégzéséhez.
- *Elérhető*: *(T/F)*, értéke *TRUE*, ha az adott eszköz már az akvarista birtokában van.

A fenti sémákat megadó CLIPS kód részletek a *8.2-es számú függelékben* láthatóak.

5.4 A rendszerben használt szabályok

Az AquAdvice rendszer szabályai hét fő csoportra oszthatóak. Az első hat kategóriába tartoznak azok a szabályok, melyek a tudásbázis közvetlen feldolgozásából származnak. A hetedik kategóriába különböző technikai jellegű szabályok találhatóak, melyek feladata a felhasználóval történő interakció irányítása, illetve a rendszer által szolgáltatott eredmények megjelenítése. Utóbbiakból lényegesen kevesebb használatára volt szükség. Az egyes szabályokra a *8.3-as függelékben* látható egy-egy CLIPS példa a forráskódból.

5.4.1 Elvégzendő részfeladatok

A szabály lényege, hogy ha adott egy *X* folyamat, mely állapota „*folyamatban*” – azaz a felhasználó célja ennek elvégzése – akkor, ezen *X* feladathoz tartozó részfeladatokat is el kell végeznie. A szabály a következőképp néz ki:

IF *feladat.név = X* **AND** *feladat.folyamatban = TRUE* **THEN**
részfeladat1.elvégzendő := TRUE,
részfeladat2.elvégzendő := TRUE,
….

5.4.2 Szükséges eszközök

Ha egy adott feladat „*folyamatban*” van, akkor a tudásbázisban hozzárendelt eszközökre szüksége van a felhasználónak, ezért azok megfelelő értékeit be kell állítani:

IF *feladat.név = X* **AND** *feladat.folyamatban = TRUE* **THEN**
eszköz1.szükséges := TRUE,
eszköz2. szükséges := TRUE,
….

5.4.3 Elvégzett feladat részfeladatai

Ha egy feladatot elvégeztünk, azaz „*elvégezve*” tulajdonsága *TRUE* értékű, akkor értelemszerűen ez azt jelenti, hogy a hozzátartozó részfeladatokat is elvégeztük.

IF *feladat.név = X* **AND** *feladat.elvégezve = TRUE* **THEN**
részfeladat1.elvégezve := TRUE,
részfeladat2. elvégezve := TRUE,
….

5.4.4 Elvégzett feladathoz tartozó eszközök

Ha egy feladatot sikeresen befejeztünk, akkor ez csak úgy lehetséges, hogy az annak elvégzéséhez szükséges eszközökkel már rendelkezünk.

IF *feladat.név = X* **AND** *feladat.elvégezve = TRUE* **THEN**
eszköz1.elérhető := TRUE,
eszköz2. elérhető := TRUE,
….

5.4.5 Előfeltételek

Bizonyos feladatok csak akkor lehetnek folyamatban, ha előzőleg már az előfeltételeihez tartozó feladatokat elvégeztük.

IF *feladat.név=X* **AND** *feladat.folyamatban = TRUE* **THEN**
előfeltétel1.elvégezve:= TRUE,
előfeltétel2. elvégezve := TRUE,
….

5.4.6 Elvégzett részfeladatok

Ha egy feladathoz tartozó összes részfeladatot elvégeztünk, akkor maga a fő feladat is el van végezve.

IF *feladat.név = X* **AND** *részfeladat1.elvégezve = TRUE*
AND *részfeladat2.elvégezve = TRUE*
….
THEN *főfeladat.elvégezve := TRUE*

6. Összegzés

A diplomamunkám elkezdésekor kialakult bennem egy konkrét kép, arról, hogy mit is fog pontosan elvégezni az általam elkészítendő szakértő rendszer. Ahogy viszont egyre mélyebben kezdtem el foglalkozni a mesterséges intelligencia ezen területével, úgy változtak az elképzeléseim is. Számos rendszert próbáltam ki, melyekben az első pillanatban a tökéletes eszközt láttam a saját programom elkészítéséhez, viszont egy kivételével mindegyik lehetőségei kevésnek bizonyultak számomra. Bár a végleges tudásbázis megalkotásában ezek az eszközök nem játszottak kézzelfogható szerepet, mégis nagyban elősegítették a későbbi munkámat, hiszen számos tapasztalatot szereztem a használatuk közben és – még ha tudat alatt is – rendszereztem az akvarisztika területén szerzett ismereteimet, így megkönnyítve azok későbbi eltárolását.

A kivételt természetesen a PCPACK jelentette, mely szinte korlátlan lehetőségeket biztosít a tudásmérnök számára, legyen szó magáról a tudásbeszerzésről, vagy annak megjelenítéséről, esetleges bővítéséről. Bár diplomamunkám során korántsem aknáztam ki minden benne rejlő lehetőséget, remélem sikerült egy kisebb áttekintést nyújtani az általa nyújtott eszközrendszerről.

A tudásbeszerzés közben szükségem volt a beszerzésre szánt ismeretek szűkítésére, az összefüggések egyszerűsítésére, mely az eldobható prototípust is eredményezte. A későbbiekben viszont beláttam, hogy erre feltétlen szükség volt, hiszen még így is egy olyan bonyolult kapcsolatgráfot kaptam, mely teljes valójában nem áttekinthető a monitoron.

A CLIPS program elkészítését rendkívüli módon megkönnyítette az ismeretek ilyen módon történt ábrázolása, könnyen és gyorsan tudtam előállítani a rendszer magját képező szabályokat. Bár a forráskód elég hosszúra sikerült, azt hiszem mégis áttekinthető és könnyen bővíthető újabb szabályokkal. A végeredményül kapott programban nincs minden funkció megvalósítva, amelyeket korábban terveztem, aminek oka a fentebb említett, így is nagyra sikerült tudásbázis. A rendszerben implementált 165 ténnyel és 211 szabállyal viszont úgy érzem, hogy sikerült egy összetett tudásalapú rendszer elkészítését bemutatnom.

7. Irodalomjegyzék

Dr. Bognár Katalin: Ismeretábrázolás és következtetés OO rendszerekben, mobiDIÁK könyvtár, 2004
http://iam035.inf.unideb.hu/mobidiak/filedownload.mobi?fid=261

Dr. Bognár Katalin: Mesterséges Intelligencia 4, egyetemi jegyzet
http://www.inf.unideb.hu/~bognar/mestint4/mi4.ps

Aszalós László: Mesterséges intelligencia közgazdászoknak, mobiDIÁK könyvtár, 2005
http://www.inf.unideb.hu/~aszalos/diak/mik/mik.pdf

Tomas Eric Nordlander - AI Surveying: Artificial Intelligence In Business
http://www.
4c.ucc.ie/web/upload/publications/mastersThesis/Artificial_Intelligence_in_Business.pdf

Dr. Dobrowiecki Tadeusz, Mészáros Tamás – A mesterséges intelligencia új területei: intelligens ágensek
http://home.mit.bme.hu/~meszaros/me/pubs/agensjegyzet_1999.pdf

Professor George F. Luger: AI History And Applications
http://www.cs.unm.edu/%7Eluger/Chap1final.htm

PCPACK5 User Manual (a telepítőcsomag része)
http://www.pcpack.co.uk/PCPACK5Download.aspx

Upgrade Magazin, 2002 Október: Artificial Intelligence Technology with a Future
http://www.upgrade-cepis.org/issues/2002/5/upgrade-vIII-5.pdf

P-H. Speel, A. Th. Schreiber, W. van Joolingen, G. van Heijst, G.J. Beijer: Conceptual Modelling for Knowledge-Based Systems
http://www.cs.vu.nl/~guus/papers/Speel01a.pdf

Bruce G. Buchanan and Edward H. Shortliffe: Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming projekt
http://www.aaaipress.org/Classic/Buchanan/buchanan.html

Pasaréti Gyula, Pethő Pál Zoltán, Illés Csaba: Akvarisztika kezdőknek és haladóknak
Szalay Könyvkiadó, 2003

Horn Péter, Zsilinszky Sándor: Akvarisztika
Mezőgazda Kiadó, 2005

Fundamentals Of Expert Systems
http://media.wiley.com/product_data/excerpt/18/04712933/0471293318.pdf

Regina Barzilayy, Daryl McCullough, Owen Rambow, Jonathan DeCristofaroz, Tanya Korelsky, Benoit Lavoie: A New Approach To Expert System Explanations
http://www.cogentex.com/papers/explanation-iwnlg98.pdf

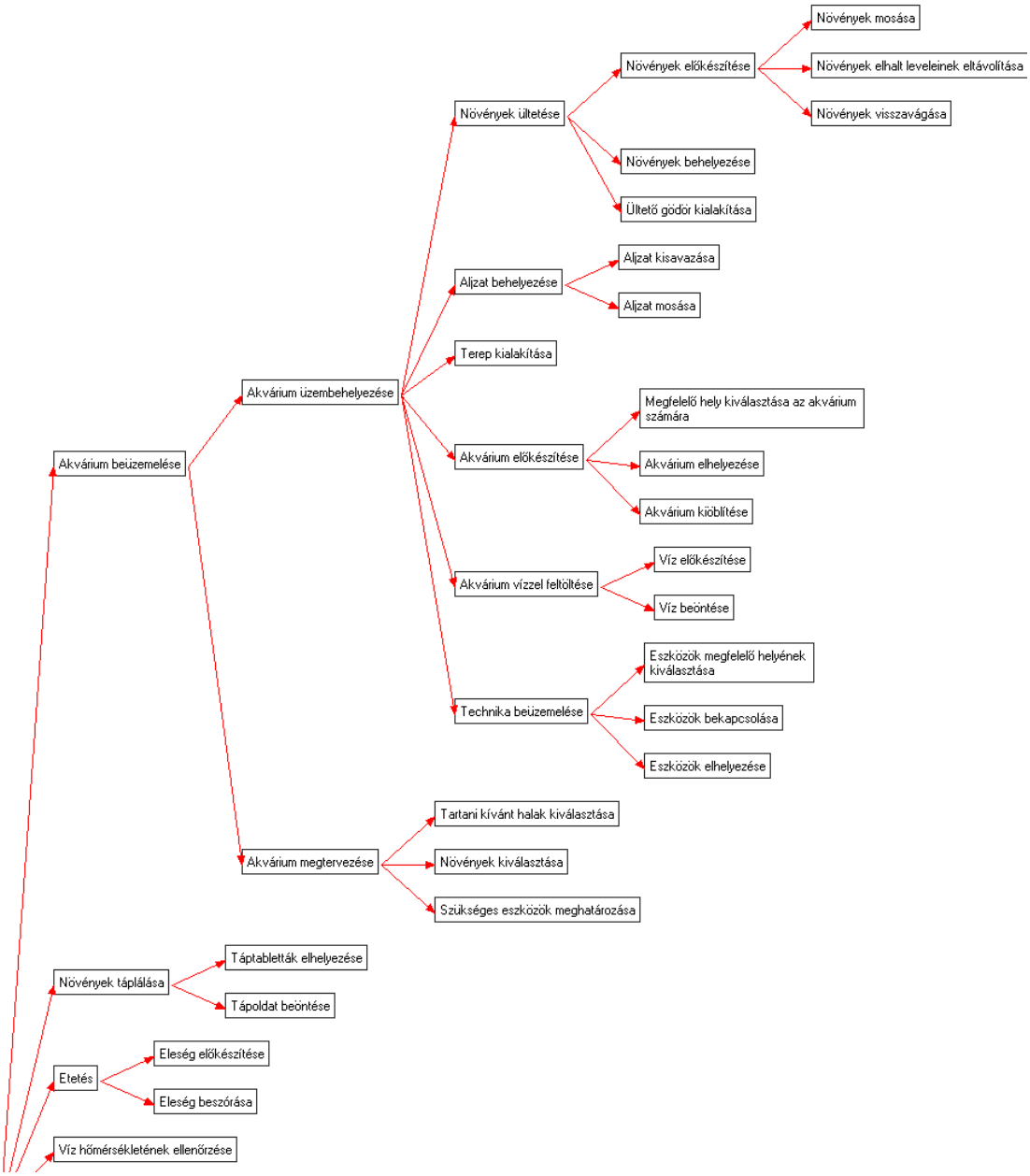
John McCarthy: Some Expert System Need Common Sense
http://www-formal.stanford.edu/jmc/someneed.pdf

Ovidiu S. Noran: The Evolution of Expert Systems
http://www.cit.gu.edu.au/~noran/Docs/ES-Evolution.pdf

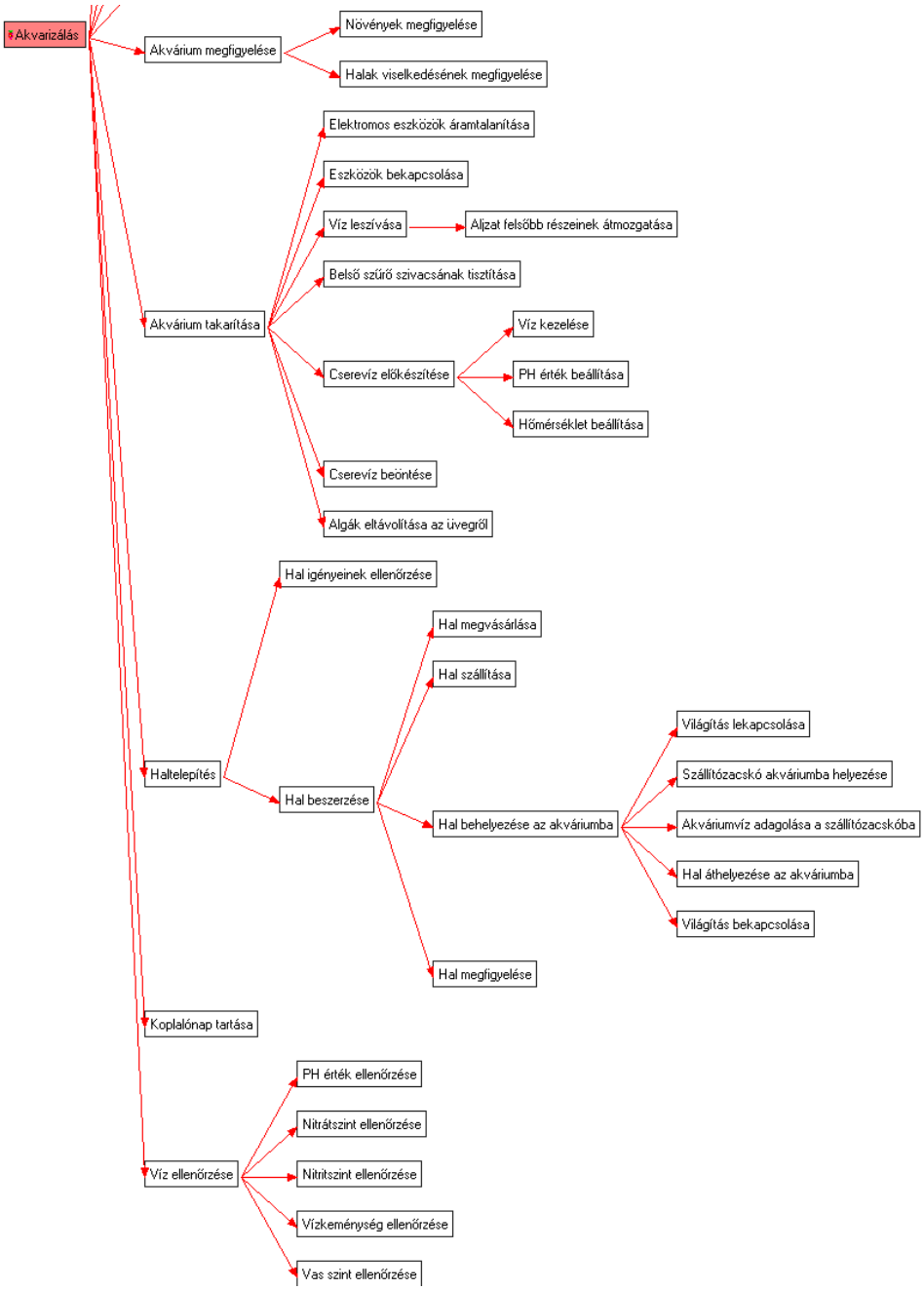
CLIPS User's Guide és Basic Programming Guide
http://www.ghg.net/clips/download/documentation/usrguide.pdf
http://www.ghg.net/clips/download/documentation/bpg.pdf

8. Függelék

8.1 Laddering technika



Laddering technika 1. rész



Laddering technika 2. rész

8.2 CLISP kódrészlet

```
(deftemplate feladat
 (slot megnevezes (type SYMBOL))
 (slot gyakorisag (type SYMBOL)
        (allowed-symbols ALKALMI NAPONTA HETENTE HAVONTA)
                (default ALKALMI))
 (slot elvegzendo (type SYMBOL)
        (allowed-symbols TRUE FALSE) (default FALSE))
 (slot elvegezve (type SYMBOL)
        (allowed-symbols TRUE FALSE) (default FALSE))
 (slot folyamatban (type SYMBOL)
        (allowed-symbols TRUE FALSE) (default FALSE))
)

(deftemplate eszkoz
 (slot megnevezes (type SYMBOL))
 (slot kategoria (type SYMBOL)
)
 (slot szukseges (type SYMBOL) (allowed-symbols TRUE FALSE) (default FALSE))
 (slot elerheto (type SYMBOL) (allowed-symbols TRUE FALSE) (default FALSE))
)
```

8.3 CLISP szabályok

```
#1
(defrule etetes-reszei
  ?fofeladat <- (feladat (megnevezes etetes) (elvegzendo FALSE) (folyamatban TRUE)
                  (elvegezve FALSE) (folyamatok-feldolgozva FALSE))
  ?reszfeladat1 <- (feladat (megnevezes eleseg-elokeszitese) )
  ?reszfeladat2 <- (feladat (megnevezes eleseg-beszorasa) )
=>
  (modify ?reszfeladat1 (elvegzendo TRUE) (sorrend 1))
  (modify ?reszfeladat2 (elvegzendo TRUE) (sorrend 2))
  (modify ?fofeladat (folyamatok-feldolgozva TRUE))
)

#2
(defrule aljzat-mosasa-eszkozok
```

```
 ?fofeladat <- (feladat (megnevezes aljzat-mosasa) (elvegzendo FALSE)
                        (folyamatban     TRUE) (elvegezve FALSE) (eszkozok-feldolgozva FALSE))
 ?szukseges-eszkoz1 <- (eszkoz (megnevezes csapviz))
=>
 (modify ?szukseges-eszkoz1 (szukseges TRUE))
 (modify ?fofeladat (eszkozok-feldolgozva TRUE))
)

#3
(defrule aljzat-behelyezese-reszei-elvegezve
 ?fofeladat <- (feladat (megnevezes aljzat-behelyezese) (elvegzendo FALSE)
                        (folyamatban FALSE) (elvegezve TRUE) (folyamatok-feldolgozva FALSE))
 ?reszfeladat1 <- (feladat (megnevezes aljzat-kisavazasa) )
 ?reszfeladat2 <- (feladat (megnevezes aljzat-mosasa) )
=>
 (modify ?reszfeladat1 (elvegzendo FALSE) (folyamatban FALSE) (elvegezve TRUE))
 (modify ?reszfeladat2 (elvegzendo FALSE) (folyamatban FALSE) (elvegezve TRUE))
 (modify ?fofeladat (folyamatok-feldolgozva TRUE))
)

#4
(defrule novenyek-mosasa-eszkozok-elerheto
 ?fofeladat <- (feladat (megnevezes novenyek-mosasa) (elvegzendo FALSE)
                        (folyamatban FALSE) (elvegezve TRUE) (eszkozok-feldolgozva FALSE))
 ?szukseges-eszkoz1 <- (eszkoz (megnevezes vodor))
 ?szukseges-eszkoz2 <- (eszkoz (megnevezes csapviz))
=>
 (modify ?szukseges-eszkoz1 (elerheto TRUE))
 (modify ?szukseges-eszkoz2 (elerheto TRUE))
 (modify ?fofeladat (eszkozok-feldolgozva TRUE))
)

#5
(defrule terep-kialakitasa-elofeltetel
 ?fofeladat <- (feladat (megnevezes terep-kialakitasa) (elvegzendo FALSE)
                        (folyamatban TRUE) (elvegezve FALSE) (elofeltetelek-feldolgozva FALSE))
 ?elofeltetel1 <- (feladat (megnevezes aljzat-behelyezese))
 ?elofeltetel2 <- (feladat (megnevezes akvarium-elokeszitese))
=>
 (modify ?elofeltetel1 (elvegzendo FALSE) (folyamatban FALSE) (elvegezve TRUE))
 (modify ?elofeltetel2 (elvegzendo FALSE) (folyamatban FALSE) (elvegezve TRUE))
 (modify ?fofeladat  (elofeltetelek-feldolgozva TRUE))
)

#6
(defrule akvarium-vizzel-feltoltese-fo-elvegezve
 ?fofeladat <- (feladat (megnevezes akvarium-vizzel-feltoltese)(elvegezve FALSE))
                        (feladat (megnevezes viz-elokeszitese) (elvegezve TRUE))
 (feladat (megnevezes viz-beontese) (elvegezve TRUE))
=>
 (modify ?fofeladat (elvegezve TRUE))
)
```