A hazai eszközfejlesztések helyzete

Domán András +, Koch Péter + +, Sántáné-Tóth Edit + + +

\+ *MULTILOGIC*
\+ + *SZÁMALK*
\+ + + *SZKI*

Tartalom

Kulcsszavak:

szakértő rendszer,
tudás- vagy ismeretreprezentáció,
keretrendszer vagy szakértőrendszer-keret,
mesterséges intelligencia.

Bevezető

Ebben a fejezetben először röviden áttekintjük a szakértőrendszer-keret piac helyzetét és ennek tükrében a hazai lehetőségeket, majd négy hazai fejlesztésű, PROLOG alapú keretrendszert ismertetünk. Az irodalomjegyzék tartalmaz néhány olyan dolgozatot is, melyeket e témakörben alaposabban tájékozódni kívánó olvasók figyelmébe ajánlunk.

1. Szakértőrendszer-keret körkép

A szakértő rendszerek (SZR) és szakértőrendszer-keretek (SZRK) fejlesztése a 80-as évek elejéhez viszonyítva jelentős fejlődésen ment keresztül napjainkig. Ennek legfontosabb eredménye az új technika kilépése a kisérleti laboratóriumokból a gyakorlati felhasználás területére. Itt csak a legjelentősebb változásokat emeljük ki, a részletes elemzéssel kapcsolatban [Koch,1987]-re utalunk.

A 80-as évek elejét a mesterséges intelligencia (MI) fejlesztők saját tapasztalatai alapján létrehozott szakértő rendszerek jellemezték. Ezek nagyméretű, egyedi alkalmazások voltak, amelyeket többnyire LISP-ben írtak és valamilyen MI specifikus gépen (Xerox 11xx, Texas Instruments, SYMBOLICS, stb.), ritkábban nagygépen futtattak. Kevés kivételtől eltekintve ezek a rendszerek nem kerültek valódi kereskedelmi forgalomba. Általában egy-egy tőkeerős nagyvállalat vagy állami szervezet (BOEING, General Motors, NASA stb) vette meg azzal a szándékkal, hogy megvizsgálja az új technológia alkalmazási lehetőségeit saját területén belül. Ezeknek a rendszereknek a használata során számos tapasztalat és eredmény halmozódott fel mind az alkalmazók, mind pedig az MI fejlesztők oldalán.

Ennek alapján egyfelől a felhasználók már ki tudtak jelölni a saját tevékenységükön belül néhány olyan területet, ahol a SZR-ek alkalmazásában kezdeti sikerek mutatkoztak, másfelől pontosabban meg tudták fogalmazni a fejlesztők számára az eszközökkel kapcsolatos igényeiket. A fejlesztői oldalon a felhasználók visszajelzései, az MI technikában ill. a hardver-szoftver feltételekben bekövetkezett fejlődés az MI eszközökben is jelentős változást eredményezett. A nagyteljesítményű munkaállomások és a PC-k megjelenésével az MI eszközök rohamosan terjedtek el szinte a teljes hardver-szoftver skálán (az MI gépektől a PC-ig). Ezzel párhuzamosan az MI fejlesztések területén korábban egyeduralkodó LISP és PROLOG mellett a hagyományos programozási nyelvek - főleg a C nyelv - is bekerültek a szakértő rendszerek fejlesztésének alapnyelvei közé. Ennek következtében beindult az egyszerűbb MI/SZR eszközök "gyártása" és "fogyasztása", azaz a MI/SZR piac kialakulása. A megnövekedett alkalmazói igények középpontjában a gyors, kényelmes és programozói ismereteket nem igénylő SZR fejlesztést kínáló eszközök, a szakértőrendszer-keretek (expert system shell), röviden keretrendszerek álltak.

A keretrendszerek (ld. pl. [Sántáné-Tóth, 1988]) a SZR-ek tipikus elemeit "előregyártva", általában a munkát támogató hagyományos segédeszközökkel (trace, grafika, on-line

help, stb.) integrálva tartalmazzák. Noha a 70-es évek végén már felmerült egy "üres" szakértő rendszer gondolata, és az első ilyen rendszert (EMYCIN) el is készítették, azonban a keretrendszerek igazi elterjedése a 80-as évek elejétől indult meg. Mára a keretrendszer fejlesztés az MI piac legdinamikusabban fejlődő komponensévé vált [Koch,1987] és - az alkalmazások húzó hatására - az implementációs nyelv és a hardver vonatkozásában széles tartományt fed le. A több tucat kereskedelmi forgalomban levő keretrendszer az alábbi három fő csoportba sorolható:

- nagyméretű, hibrid (többféle ismeretreprezentációs technikát használó) rendszerek, amelyek többnyire LISP-ben íródtak és MI specifikus gépen futnak (ART, LOOPS, KEE, KNOWLEDGE CRAFT stb.),
- közepes méretű, általában homogén - főleg szabályalapú és/vagy objektum-orientált technikát alkalmazó - eszközök, amelyek többségét C-ben(!) írták és különbözö (nem MI specifikus) munkaállomáson (MicroVAX, SUN, APOLLO stb.) implementálták (NEXPERT, KES II, HUMBLE, GURU stb.),
- kisméretű, PC-orientált eszközök, szabályalapú technikával általában hagyományos nyelven (C, PASCAL, stb.), néha valamelyik PC-s PROLOG-ban (Arity, Turbo) implementálva, IBM PC kompatibilis gépeken (INSIGHT, EXSYS, M.1, KDS, ESP ADVISOR, EXPERT-EASE, stb.).

A közepes és a PC-orientált keretrendszerek megjelenése és elterjedése a felhasználók széles köre számára hozzáférhetővé tette az egyszerűbb SZR technikákat. Ennek, és a korábbi sikeres SZR alkalmazásoknak a hatására számos vállalat (főleg az USA-ban) döntött a szakértő rendszerek gyakorlati bevezetése mellett (IBM, BOEING, GENERAL MOTORS, DuPont, stb.). A gyakorlati megvalósítás során a vállalatok lényegében az alábbi négy fő stratégiai irányvonalba rendeződtek (részletesebben l. pl. [Koch,1987], vagy a jelen kötetben [Sántáné-Tóth,1988]):

stratégia 1:
nagyméretű, egyedi szakértő rendszerek fejlesztése,

stratégia 2:
kis- vagy közepes méretű alkalmazások fejlesztése
a már meglevő, munkaállomáson vagy PC-n működő
rendszerek támogatására,

stratégia 3:

konzultáció és fejlesztési támogatás a SZR-eket használni akaró végfelhasználók számára,

stratégia 4:

az SZR technika felhasználása a hagyományos nagygépes alkalmazások támogatására.

Ami a hazai szakértő rendszer és ezzel szoros összefüggésben a keretrendszer fejlesztést illeti, a következőket mondhatjuk. Mivel egy itthon, az alkalmazói körökben kevéssé ismert technikáról van szó, olyan megközelítést kell keresni, amely a felhasználók részéről a legkisebb befektetést és kockázatvállalást jelenti, és viszonylag rövid idő alatt (0.5-1.5 év) eredményeket produkál. A fenti stratégiák közül a 2-es és 3- as számú adaptálása látszik a legcélszerűbbnek. Ennek megfelelően, tekintettel az IBM PC alkalmazók nagy számára, első lépésben arra kell törekedni, hogy a már működő PC-s alkalmazások közül megkeressük azokat, ahol (saját tapasztalataink és a nyugati piaci információk alapján) a SZR alkalmazása várhatóan kimutatható haszonnal jár. Az első keretrendszer fejlesztések a kisméretű, de a hazai viszonyok között elterjedt szoftverrel, valamint a hagyományos nyelvekkel integrálható SZR-ek fejlesztését támogató eszközök előállítását kell megcélozzák, kiegészítve azokat a felhasználó kényelmét szolgáló segédeszközökkel (színes, sokablakos képernyőtechnika, on-line help, grafika stb.). Önmagában azonban a jó keretrendszer nem garantálja a sikeres SZR fejlesztést. Ehhez a kérdéses keretrendszer ismeretén kívül a „tudás" kinyerésének, formalizálásának és célszerű működtetésének a technikáját is ismerni kell. Ez azt jelenti, hogy a fejlesztést megfelelő oktatási-konzultációs tevékenységgel kell összekötni, és ki kell építeni az MI fejlesztő és a végfelhasználó között "közvetítő" specialisták, ismerettechnológusok ("knowledge engineer"-ek) rétegét. A SZR-ek hazai sikere nemcsak a megfelelő fejlesztő eszközök meglététől, hanem ennek a rétegnek a mielőbbi kialakulásától is erősen függ.

Minden eszköz - legyen az SZRK is - azonban csak úgy használható fel eredményesen, ha előnyeit és korlátait egyaránt ismerjük. Az alábbiakban erről lesz szó.

A PC-s keretrendszerek jól alkalmazhatók a gyakorlatban is, megfelelően kiválasztott kis- és közepes méretű SZR-ek építésére. Emellett előnyösen használhatók fel az új, "ismeretalapú" fejlesztési technológia elsajátításához is. Ezen azt kell érteni, hogy amennyiben

- a megoldandó feladat tárgyköri ismeretei leirhatók a rendszer által támogatott ismeretábrázolási formalizmussal, és
- az így kialakìtott ismeretbázis meghajtható a rendszer következtető mechanizmusaival,

akkor a rendszerépìtésben - különösen pedig a gyors prototípuskészìtésben - hatékony segìtséget tud nyújtani az adott keretrendszer.

A jelenlegi keretrendszereknek - főleg azok PC-s képviselőinek - vannak azonban jelentős korlátai is (itt nem arra gondolunk, hogy pl. nem tudnak tapasztalataikból tanulni, egysíkúan ábrázolják a valóságot, stb.). Saját tapasztalataink is igazolják, hogy egy közepes méretű feladat leírásához vagy nem elegendő a PC-n rendelkezésre álló tár, vagy ha igen, olyannyira lelassul a megoldás keresése, hogy a végterméknél már nem elfogadható a sebesség. A jelenlegi keretrendszerek általában nem biztosìtanak kompatibilitást se oldalirányba, se felfelé (vagyis azonos ill. magasabb gépkategórián működő keretrendszerek irányba). Ilyen probléma esetén

- vagy újraépìtjük ismeretbázisunkat valamely magasszintű programozási nyelven - a végrehajtási mechanizmust, a magyarázatadást és a felhasználói interfészt is beprogramozva (ami egyetlen rendszer esetén nem rentábilis),
- vagy veszünk egy megfelelőbb eszközkészletet, megtanuljuk annak használatát, és újra felépìtjük (most már sokkal gyorsabban) az ismeretbázist.

Nagyobb a probléma akkor, ha fejlesztés közben döbbenünk rá, hogy a keretrendszer által nyújtott ismeretábrázolási mód nem alkalmas a feladat leírására. Sok feladat ugyanis struktúrált objektumok, fogalmak hierarchikus rendszerével írható le igazán, melynél bizonyos objektum-tulajdonságok még öröklődhetnek is. A legtöbb PC-s keretrendszer azonban csak egyféle (többnyire szabályalapú) ismeretábrázolást támogat. Ebből a szempontból előnyben vannak a nagyobb gépkategóriákon működő olyan kereskedelmi keretrendszerek, melyek több ábrázolási módot támogatnak (hibrid rendszerek). A jövő útja - remélhetően - Magyarországon is az, hogy munkaállomások és nagyobb gépek megjelenésével bővül az elérhető SZR épìtő eszközök "ereje" és választéka. Az új, ismeretalapú technológia elterjesztésében és oktatásában azonban sokáig jelentős szerepük lesz még a PC-s keretrendszereknek.

Az alábbiakban röviden ismertetünk három hazai fejlesztésű szakértőrendszer-keretet (GENESYS, ALL-EX, MPROLOG Shell), és beszámolunk az MPROLOG alapú alkalmazá-

sokat támogató moduláris eszközfejlesztés első eredményeiről (pl. egy párbeszéd megvalósìtását támogató keretrendszerről, a DIALOG-ról). A három SZRK szabály- ill logikai alapú ismeretreprezentációt támogat; mind a négy keretrendszer fejlesztése PROLOG alapokról indult ki. Ezek az első eredményei a általános célú, hazai SZR- eszközfejlesztéseknek.

2. Az ALL-EX szakértőrendszer-keret

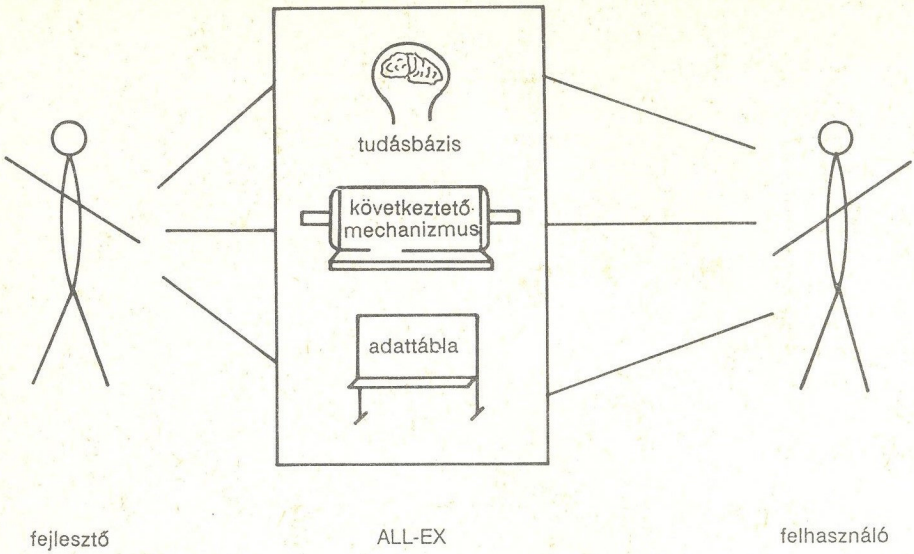
2.1. Bevezetés

Az ALL-EX amely az ALL kutató fejlesztő kisszövetkezetben készült, egy hosszabb fejlesztési projekt első elkészült eredménye, a ismeret-alapú rendszerek fontosabb funkcióit tartalmazó szabály-alapú, visszafelé következtető rendszer: IBM PC környezetben működik. A jelenlegi fejlesztés célja az volt, hogy egyszerű, de mégis sokféle területen alkalmazható rendszert hozzunk létre úgy, hogy ezt ne csak a mesterséges intelligencia művelői használhassák, hanem egy-egy konkrét terület szakemberei is, kialakítva saját szakértő, tanácsadó rendszerüket. A fejlesztésben és terjesztésben megpróbáltunk olyan feltételeket kialakítani, hogy az ALL-EX minél több felhasználóhoz eljuthasson, amivel elősegítenénk a hazai megkésett fejlődést a számítógépes szakértő rendszerek megismerésében, alkalmazásában. Az egyszerűség mellett azonban arra is törekedtünk, hogy speciális alkalmazásokhoz - egyenlőre opcióként - különlegességeket is nyújtson a keret. Ilyen különlegességnek számít a szimulációs modellezés integrálása (deep reasoning) a rendszerbe. A fejlesztési projekt távolabbi céljai között szerepel egy többszintű tudásfeldolgozó rendszer, többféle tudásreprezentációs paradigmával (frame-bázisú, objektum- orientált), többféle következtetési stratégiával, automatikus tudásösszegyűjtő- és elemző rendszerrel és kifinomultabb felhasználói interfésszel. Ezek a fejlesztések már nem korlátozódnak az IBM PC szintű számítási környezetre.

2.2. Általános áttekintés

Az ALL-EX Prolog-alapú szakértőrendszer-keret, segítségével mikroszámítógépes környezetben tudásalapú rendszerek építhetők és futtathatók önálló rendszerként is. Az ALL-EX szabályalapú rendszer: a szakértő tapasztalati tudása vagy más, előírásokból szerzett ismeretek, egy kölcsönösen független HA-AKKOR szabályokba szervezett tudásbázisba kerülnek. A rendszer a felhasználóval folytatott konzultáció során következtetéseket von le, tanácsot ad az adott problémákra vonatkozólag. A konzultáció közben és utána is magyarázatot lehet kérni a következtetésre vagy az adott kérdésre vonatkozólag. Az ALL-EX felépítése igen egyszerű, három fontos komponenst tartalmaz: a tudásbázist (szabályok és tények), a következtető mechanizmust és az adattáblát (közbenső következtetések eredményei). Mindezekhez az elemekhez hozzáférhet a rendszer fejlesztője (knowledge engineer) csakúgy mint a felhasználó (bár mindkettő

más-más célból). Ezekre a hozzáférésekre az ALL-EX külön interaktív környezetet biztosit.



2.1. ábra: Az ALL-EX felépítése

Az ALL-EX további lehetőségei, szolgáltatásai: a tudásbázis interaktív nyomonkövetése és módosítása, automatikus (default) kérdés generálása, válasz-helyesség ellenőrzés, bizonytalan tudás kezelése, opcionális hozzáférés dBASEIII adatbázisokhoz, opcionális interfész a PROLOG-hoz, ablakok és menük használata, stb.

2.3. Az ALL-EX felépitése

Tudásbázis

Az ALL-EX, mint a szakértő rendszerek többsége, szabályalapú rendszer, ennek előnyei ismertek. Egyszerű és kényelmes a tudás formalizálása, könnyű magyarázatadó rendszert készíteni, továbbá a tudásbázis bővítése a meglévők változtatása nélkül is lehetséges. A tudásbázis a szabályok mellett tényeket és metatényeket tartalmaz. Ez utóbbiak a problémamegoldás menetét befolyásolják. Az egyes elemek leírását egyszerű szintaxisú nyelven kell megadni.

Példák:

Szabály-5: ha kiír("Egyszerű szakértő rendszer kiválasztása")
és tudásleírás = szabályalapú
és következtető_rendszer = visszafelé_következtető
és eredet = hazai
akkor szakértő_rendszer = allex bf 70.

tb_1: cél = szakértő_rendszer

tény-2: eredet = hazai

kérd_2: kérdés (tudás-leírás) = "Milyen következtető
mechanizmust használ a rendszer?"

opc_2: opciók (tudás-leírás) = [szabály-alapú,
objektum-orientált, hálós, vegyes, egyéb]

Következtető mechanizmus

Az ALL-EX tudásleírásában kulcsszerepet játszik a kifejezés, másnéven koncepció vagy paraméter. A következtetés eredménye általában egy (cél)-kifejezés vagy közbenső kifejezések értékének meghatározása - a bizonytalansági faktor (bf) figyelembevételével. Mint visszafelé következtető rendszer, az ALL-EX egy kifejezés értékét akkor határozza meg, amikor arra egy másik kifejezés kiértékelésénél van szükség. Minden egyes kifejezést az ALL-EX ténylegesen csak egyszer értékel ki, ezt követően az érték felkerül az adattáblára is.

A kiértékelés forrása így többféle lehet:

- adattábla, ha az adott kifejezés már értéket kapott
- tudásbázis ha szabályból vagy tényből levezethető
- felhasználónak feltett kérdésre kapott válasz
- ismeretlen (unknown) érték, ha a fentiek egyike sem határozta meg a kifejezés értékét.

Az ALL-EX a szokásos kétféle magyarázat-adásra képes: a "hogyan" típusú alkalmas a következtetési lánc (visszafelé) bejárására, míg a "miért" típusú választ ad a kérdésfeltevés okára.

Szimuláció

Az ALL-EX egyedi vonása a szimulációs modellezés lehetősége. A szakértő rendszer használata során lehetőség van arra, hogy egy esemény komponenseinek egyidejű (időbeli) viselkedése része legyen annak a tudásnak, amelyet a következtetés során a rendszer használ. A technikai megvalósítás PROLOG-alapú szimulációra épül: ebben autonóm, saját céllal rendelkező processzek működnek tetszőleges számban. A processzek egyike a szakértő rendszer, amely PROLOG-interfésze révén kapcsolatot tarthat a rendszer többi komponensével. A szimulációs modellezés jelentősége többrétű. Amellett, hogy a modellezésből gyűjthető tudás a rendszer tudásbázisát bővítheti (deep reasoning), alkalmas arra is, hogy egy valós pl. diagnosztizálandó rendszert helyettesítsen a szakértő rendszer számára. (Részletesebben lásd [Futó, 1988] tanulmányát, ugyanebben a kötetben)

Felhasználói interfész

Az ALL-EX korszerű menű- és ablaktechnika alkalmazásával kényelmes párbeszédes kapcsolatot biztosít a felhasználó és a rendszer között. A háromféle képernyőminta egy-egy környezetet jelent. A fejlesztési környezet elsősorban a tudásbázis kezelését szolgálja: tudásbázis, adattábla szerkesztésével, file- kezeléssel, opciók állításával, míg a konzultációs környezet a párbeszédhez ad keretet. Külön képernyő típus a magyarázó környezet, ez biztosítja a következtetési fa monitorozását (tetszőleges bejárását).

Következtetés

Az ALL-EX eddigi 10-15 felhasználójának kedvezőek a tapasztalatai, bár a tényleges alkalmazhatóság érdekében számos igény is felmerült. Ilyenek a külső (soros vonali-, külső nyelvi-, kereskedelmi szoftver-) interfész, továbbá a nagyobb tár kihasználásának és grafika alkalmazásának az igénye.

Végezetül egy általános, minden szakértőrendszert érintő kérdés: a gyakorlati alkalmazhatóság lényeges akadálya ma Magyarországon az ún. "knowledge engineering" tevékenység ill. ezzel kapcsolatos ismeret hiánya. Ezért az alkalmazások sikerét nagyban befolyásolja az, hogy ez a diszciplina mennyire terjed, milyen mértékben állnak rendelkezésre hozzáértő szakemberek.

3. A GENESYS szakértőrendszer-keret

A GENESYS nevű szakértőrendszer-keret, amely a SZÁMALK szakértői rendszerek osztályán készült a PC kategóriájú SZRK-ek közé tartozik (standard konfigurációjú IBM PC/AT és kompatibilis gépeken fut). A GENESYS elsősorban diagnosztikai tìpusú problémákat megoldó szakértő rendszerek fejlesztésére alkalmas. Az alábbiakban a rendszer főbb tulajdonságait és felépítését vázoljuk.

3.1. A GENESYS fontosabb tulajdonságai

A GENESYS főbb tulajdonságai a következők:

- szabályalapú tudásreprezentáció,
- előrefelé és hátrafelé haladó következtetés,
- magyarázatadás,
- a szabályok előfeltételeihez ún. bizonyossági tényező (CF: certainty factor) ill. súly rendelhető,
- háromféle kiértékelési stratégia választható (MYCIN-szerű, küszöbölt CF, súlyozott átlag),
- a rendszer lehetőséget ad a SZR fejlesztőnek hipotézis-hálók definiálására, és ezzel a beépìtendő szaktudás struktúrálására (opcionális),
- a rendszer biztosìtja a tudásbázis konfigurálhatóságát és több tudásbázis láncolását,
- a rendszer lehetőséget ad a felhasználónak az adatgyűjtés (kérdésfeltevés) startégiájának explicit meghatározására előre felé haladó következtetés esetében (kérdéshálók),
- lehetőség van aritmetikai műveletek és matematikai kifejezések kezelésére szabályszinten,
- dBaseIII adatállományok elérése lehetséges,
- a szabályok következmény-részében procedúra-hìvások adhatók meg,
- a rendszer könnyen kezelhető, színes sokablakos képernyőtechnikára, félgrafikus hálókezelésre és "on-line help" lehetőségekre támaszkodó felhasználói felülettel rendelkezik .

A fenti feladatok megoldására a GENESYS-en belül néhány olyan módszer is alkalmazást nyert, amelyek kissé eltérnek a PC-s keretrendszerek építésénél alkalmazott technikáktól. A továbbiakban a GENESYS felépítését vázoljuk.

3.2. A GENESYS felépítése

A GENESYS rendszer moduláris felépítésű; az egyes modulok a szakértő rendszer építésének és használatának megfelelő fázisát támogatják. A fő komponensek:

- a Tudásfelvivő Modul (TM),
  amelynek feladata a tudásbázis felvitele és karbantartása,
- a Generáló Modul (GM),
  amelynek feladata a megadott tudásbázis(ok)ból a felhasználó által specifikált szakértő rendszer generálása,
- Futtató Modul (FM),
  amelynek feladata az elkészitett SZR működtetése.

A Tudásfelvivő Modul

A GENESYS rendszer az alábbi tudáselemekkel dolgozik:

- szabályok,
- objektumok,
- hipotézisek,
- hipotézisháló(k) (opcionális),
- kérdésháló(k).

A fentiek közül a hipotézis- ill. a kérdéshálók alkalmazása a GENESYS-re jellemző módszer, amely más PC-s keretrendszerekben nem található meg. Az előbbi a probléma-taxonómia (pl. betegségek összefüggésrendszere) definiálását és a tudásbázis particionálását, az utóbbi a helyes kérdezési (adatgyűjtési) stratégia felépìtését teszi lehetővé. A tudásbázis particionálása (különösen a PC-s SZRK-ek esetében) fontos hatékonyságnövelő-eszköz, hiszen ìgy lényegesen nagyobb tudásbázisból "faragható le" az aktuálisan szükséges, a memóriában tárolandó tudásmennyiség.

A tudásbázis láncolás a fenti particionálással együtt lehetővé teszi a több kisebb méretű (100-200 szabályból álló) tudásbázis feletti navigálást és ezzel növeli a szakértő rendszer hatékonyságát. A kérdésháló megfelelő felépítésével pedig biztosítható az, hogy a konzultáció során a kérdésfeltevés ne szekvenciálisan, hanem minden ponton az addig begyűjtött adatoktól (azaz, a konkrét helyzettől) függően haladjon tovább. A szabályok, objektumok és hipotézisek a GENESYS-ben is hasonló szerepet játszanak mint az egyéb keretrendszerekben, vagyis a tapasztalati tudás, a jelenségek és az igazolandó állìtások leírására szolgálnak: erről részletesebben [GENESYS, 1987] ad tájékoztatást.

A GENESYS-ben a tudásbázis minden eleme módosítható, törölhető ill.új elem vihető fel a TM segìtségével.A rendszer minden módosìtásnál bizonyos ellenőrzéseket végez el (pl. hurokellenőrzés a hálók esetében), ezzel biztosìtja a TB konzisztenciáját a változtatások során.

A Generáló Modul

A Generáló Modul (GM) feladata az, hogy a megadott azonosìtójú tudásbázisokból a felhasználó utasìtásai szerint felépìtse a szakértő rendszert. A tudásbázis azonosìtója, a kiválasztott következtetési stratégia (forward, vagy backward chaining) ill. néhány szöveges ìnformáció alapján a GM automatikusan állìtja össze a szakértő rendszer szükséges elemeit a Futtató Modul számára [GENESYS, 1987].

A Futtató Modul

A Futtató Modul (FM) feladata a generált szakértő rendszer működtetése a végfelhasználónál (azaz ott, ahol a szakértelemre szükség van). A GENESYS-szel "legyártott" SZR természetesen a fejlesztő környezettől függetlenül futtatható, de a végfelhasználó nem módosìtja. A generáláskor választott stratégiától függően backward vagy forward módon működik, a kiértékelt hipotézisekhez megbìzhatósági faktort rendel, a konzultáció során magyarázat kérhető, és az eredmények lementhetők későbbi feldolgozás céljaira [GENESYS, 1987].

A GENESYS rendszer részletes dokumentációval, oktatási segédanyaggal, betanìtással és demonstrációs példákkal áll rendelkezésre.

Referenciahelyek:

Villamosenergiaipari Kutató Intézet,
Tolna megyei Tanácsi Kórház,
ÉTI,
Orvostovábbképző Egyetem,
ELTE Számìtástechnikai Tanszék,
JATE Kalmár László Kibernetikai Laboratórium,
COMPORGAN,
MEDICOR
MTA Ipar-és Vállalatgazdálkodási Kutató Intézet.
Borsodtávkő
Környezetvédelmi és Vízgazdálkodási Minisztérium
Magyar Állami Eötvös Lóránd Geofizikai Intézet

4. MPROLOG alapú alkalmazásokat támogató eszközkészletek, keretrendszerek

4.1. Általános áttekintés

Az MPROLOG (Modular PROLOG) az SZKI-ban kifejlesztett "mesterséges intelligencia alapnyelv" (ld. [MPROLOG,1985]). Gazdag beépìtett eljáráskészlettel, hatékony programfutást biztosìtó komponensekkel, kényelmesen használható programfejlesztő környezetttel rendelkezik. Az MPROLOG nagy előnye az, hogy több, mint tizenötféle géptìpuson (mikro-és nagygépen, munkaállomáson) érhető el, mindenütt ugyanazt a nyelvet realizálva. Ilymódon a felhasználói programok könnyen hordozhatók; IBM PC-n indìtva a fejlesztést, ha a tárméretek vagy a végrehajtási sebesség akadályozza a munkát, kevés és jól kézbentartható módosìtással át lehet térni nagyobb géptìpusra. Fordìtva, könnyen lehet nagygépes környezetben kifejlesztett alkalmazásokat kisebb géptìpusokra átvinni - minden esetben elég a forrásprogramot egy példányban karbantartani a különböző környezetben futó alkalmazásokhoz.

A nyelvi kompatibilitás ilyen szigorú megvalósìtása miatt az alkalmazói programok (ìgy SZR-ek) hordozása is viszonylag könnyen megoldható. Az MPROLOG külső környezeti kapcsolatai (pl. adatbázis- és nyelvi interfészek) is rendelkezésre állnak; ezeknek a különböző géptìpusokra való hordozásáról maga az MPROLOG rendszer gondoskodik. Az MPROLOG alapú alkalmazásokat támogató eszközkészletek, keretrendszerek alternatìv megoldásokat nyújtó, a lehető legnagyobb mértékben gépfüggetlen eszközök - MPROLOG modulok. E modulokból sokféle összetett eszközkészlet épìthető fel, ìgy keretrendszer is - a mindenkori feladatosztály, illetve a megrendelő/felhasználó igényei szerint. Az ezirányú fejlesztések már korábban beindultak. A következőkben rövid helyzetképet adunk az eddigi eredményekről, és a további fejlesztési tervekről.

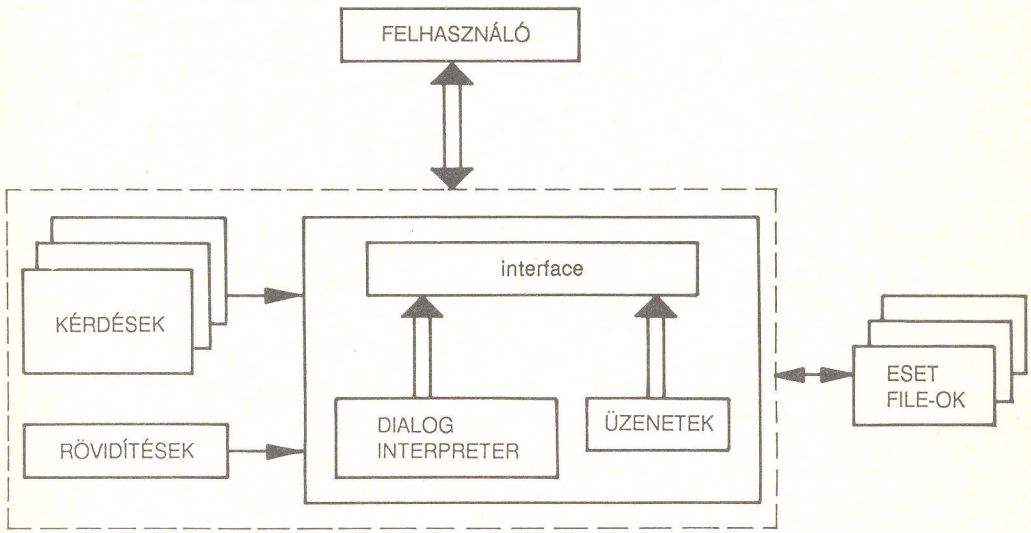
Az MPROLOG 2.3-as kiadása a következő kétféle felhasználói interfész megvalósìtását támogatja: adatbekérés előre megtervezett lépésekben (DIALOG), ill. adatbekérés igény szerint (QUERY) - ezekről a 4.1 és 4.2 pontokban lesz szó. Tervbe van véve egy harmadik féle interfész biztosìtása is: különböző adatnyilvántartó rendszerek és egyéb - pl. szakértő - rendszerek által létesìtett adatállományokból történő rugalmas adatkinyerés lehetősége (igény szerint adatmódosìtással egybekötve).

Rendelkezésre áll egy modulkészlet MPROLOG programok futásközi állapotának megjelenìtésére (TREE-TYPE), melyet a 4.3 pontban vázolunk. A 4.4 pont az előző modulok-

ból felépìtett, célvezérelt végrehajtást támogató SZRK (MPROLOG Shell) első változatának rövid ismertetését tartalmazza. A következőkben az alkalmazott fejlesztési technikák bemutatására koncentrálunk.

4.2. MPROLOG DIALOG: párbeszéd szervező keretrendszer

Az MPROLOG DIALOG párbeszédszervező keretrendszer az SZKI-ban történt korábbi MPROLOG alkalmazások igényei, valamint a számìtógéppel támogatott oktató programok szerzői nyelveinek figyelembevételével fejlődött ki ([Ecsedi-Tóth és Sántáné-Tóth,1987]); a rendszer kézikönyve a [DIALOG,1988]. A [Sándor,1987] dolgozat a keretrendszer első sikeres alkalmazását mutatja be. Az 1. ábrán a rendszer funkcionális felépìtése látható.



4.1. ábra. Az MPROLÓG DIALOG rendszer funkcionális felépítése.

A rendszer a felhasználó és a gép közötti párbeszédet olyan irányìtott gráffal reprezentálja, melynek előre megadott bejárási útjait a párbeszéd során, a mindenkori szituációhoz alkalmazkodva, "intelligens" módon lehet bejárni. A DIALOG ezenfelül lehetőséget

biztosìt a párbeszéd tagolására is - ekkor a rendszer adott részgráfok bejárásával foglalkozik.

A gráf csúcspontjaiban tárolt információk egyrészt az egyes párbeszéd-elemeket (KÉRDÉSEKet, a válaszadáshoz RÖVIDITÉSEKet, stb.), másrészt azok sorrendjét írják le - beleértve a párbeszéd dinamikus irányìtását is. Ezenfelül tetszőleges MPROLOG eljárások meghívására is mód van. A DIALOG tehát nyitott a teljes MPROLOG nyelv felé, amely a korábban emlìtett előnyökkel jár: egyrészt az esetleg csatlakozó SZR (mint MPROLOG program) "kérhető fel" valamely részfeladat megoldására, másrészt lehetőség van az MPROLOG-ból elérhető adatbázis- és magasszintű nyelvi kapcsolatok felhasználására éppúgy, mint a saját (ill. a későbbiekben külső) adatállományok rugalmas kezelésére.

A gráfbejárás tagolásán kivül lehetőség van a felhasználó válaszától függően elágaztató, továbbá ciklus- és frame- struktúrájú (kérdőívszerű) adatbekérő csúcsok használatára is. Minden egyes csúcshoz előfeltétel(eke)t lehet hozzá rendelni, amelyek teljesülése szükséges a csúcs aktivizálódásához. Így a dialógus eddigi menete során a rendszerrel közölt tárgyköri adatokkal és egyéb (pl. a csatolt SZR működési állapotát jelző) információkkal vezérelni lehet a dialógus további menetét.

További fontos tulajdonság, hogy a felhasználó válaszaihoz adatellenőrző eljárásokat is lehet rendelni. Azt is elő lehet írni, hogy milyen adatstruktúrák keletkezzenek az egyes csúcspontok által reprezentált dialógus-részlet végrehajtásakor; ezen adatstruktúrákat ki lehet menteni file-ba (ESET FILE-ba), illetve ezek segítségével fel lehet idézni (ellenőrzési, adatmódosìtási, aktualizálási céllal) korábbi párbeszédeket. Ugyanezzel a technikával biztosìtható az aktuális párbeszéd során korábban adott válaszok módosìtása is.

A DIALOG rendszert egy MPROLOG-ban írt DIALOG-INTERPRETER készlet szolgálja ki. A felhasználó a párbeszéd megkezdésekor választhat, hogy milyen üzemmódban kíván beszélgetni a rendszerrel: menü vezérelten; a válaszokat mindig begépelve (bizonyos géptìpusoknál a képernyőkezelés csak ezt az üzemmódot támogatja); esetleg korábban kimentett információk segìtségével csak fel akarja idézni azt a korábbi párbeszédet, stb. A gráf a rendszerben úgy van ábrázolva, hogy mind a párbeszéd során közlendő (tárgyterületi) szövegek, mind pedig a rendszer-ÜZENETEK modul szinten könnyen lecserélhetők (magyar, angol, német, stb. változatok könnyen készìthetők). Ehhez egy szövegcserét támogató MPROLOG segédprogramot lehet használni, majd megfelelő (forrásszintű) modulcsere után újra kell generálni a rendszert.

Egy alkalmazói párbeszéd-gráf nem más, mint egy MPROLOG modulkészlet, melyet az MPROLOG programfejlesztő alrendszere, a PDSS segìtségével kényelmesen lehet kifejleszteni (felépìteni, szerkeszteni, tesztelni, valamint megkeresni és módosìtani az össze-

tartozó struktúra-részleteket). Egy dedikált fejlesztő környezet is rendelkezésre fog állni, amely megkönnyíti a gráf csúcsait leíró frame-ek, élek, valamint a gráf részgráfokra való bontásának és inicializálásának leírásához szükséges információk bevitelét, módosítását-szerkesztését, megjelenítését (kirajzolását), verifikálását és tesztelését. (E fejlesztő rendszer természetesen kiszolgálja a következő pontban említésre kerülő QUERY interfész-szervező modulkészlet fejlesztői igényeit is.)

4.3. QUERY: párbeszédszervező modulkészlet

A QUERY párbeszédszervező modulkészlet ún. "query the user" típusú, igény felmerülésekor információt kérő párbeszédet biztosít a rendszer és a felhasználó között ([Molnár, 1987] [MPROLOG Shell,1988]). A QUERY bármely MPROLOG programhoz hozzácsatolható, és a programfutás során az MPROLOG kivételkezelő mechanizmusa hozza működésbe.

Az előző pontban használt gráfszemlélet keretében maradva, a QUERY egyes csúcspontokat kezel: egy adott időben mindig csak egyetlen csúcspontot lát, és az ahhoz rendelt (a felhasználótól adatokat bekérő) üzenetváltást bonyolítja le. A QUERY akkor éled fel az MPROLOG program futása során, ha az felhasználna egy olyan adatot (pontosabban: MPROLOG tényállítást), amely hiányzik (nincs definiálva). Ha ez az adat "kérdezhető"-ként van deklarálva, a QUERY a hozzá rendelt metadeklarációk felhasználásával lebonyolítja az adatbekérést. Amennyiben a "kérdezhetőséget" megengedő deklaráción kívül más deklaráció nem szerepel, a QUERY szabványos szöveggel saját maga oldja meg az adatbekérést (pl. "Is it true that <MPROLOG állítás>"). A "barátságos" párbeszéd megvalósításához szükséges természetes nyelvű kérdés szövegét, a választ vezérlő menűt, valamint a párbeszédet irányító és ellenőrző metadeklarációkat a programfejlesztés során bármikor, bármilyen sorrendben és egyenként is megadhajta a fejlesztő (inkrementális programfejlesztés). A párbeszédhez rendelt szövegek (kérdőmondat, esetleg menük, adatellenőrző eljárások, a kérdés feltevésének előfeltétele) a DIALOG-éhoz hasonlók.

A DIALOG és a QUERY egyaránt alkalmas a felhasználóval való párbeszéd előre megtervezett sorrendű ill. a feldolgozó program logikája által vezérelt szervezésére. Egyazon dedikált fejlesztő környezet szolgálja ki mindkettőt. E két rendszer integrálva van (egy MPROLOG programon belül a megfelelő adatstrukturákat mind a két módon működésbe lehet hozni), együttes használatukat azonban a jelenlegi PC-ken rendelkezésére álló tárméret korlátozza.

4.4. TREE-TYPE: végrehajtási fa megjelenítése

A TREE-TYPE modulkészlet MPROLOG programok dinamikus (futásközben keletkezett) állításainak megjelenítésére szolgál. A végrehajtási fa egyetlen csomópontját (azaz egy állítást, fejből és törzsből álló MPROLOG statement-et) tudja kirajzolni, illetve annak elemeiből kiindulva a fa további részleteit tudja megmutatni. A TREE-TYPE a kiválasztott állítás fejét, valamint a törzset képező feltételeket („eljárásokat") azok esetleges argumentumaikkal együtt ábrázolja, a feltételek közti "és" ill. "vagy" kapcsolatok feltüntetésével. Lehetőség van ezen ábrázolt elemek közül egynek a kijelölésére, és e kijelölt elemhez kapcsolt definícióra való átlépésre - a hívási láncban mindkét irányban (a hívó ill. a hívott eljárás felé). A modul megoldja a folytatósorok, a nagyméretű fák, valamint az MPROLOG állítások törzsében zárójellel elkülönített ún. csoportok kezelését is.

Felhasználóbarát, természetes nyelvű magyarázatot adó eljáráskészlet alakítható ki a TREE-TYPE segítségével. Ez lehetővé teszi, hogy egy állításhoz (azaz SZR-szabályhoz) metadeklaráció révén olyan természetes nyelvű magyarázatot fűzzünk, melybe beépíthetjük az állításban szereplő argumentumok aktuális értékét is ("szöveg-animáció"). Ilyen metadeklaráció a programfejlesztés folyamán bármikor megadható; a megjelenítésnél azok az állítások, melyekre még nem adtunk ilyen deklarációt, az eredeti MPROLOG definíció formájában kerülnek megjelenítésre (vö. a már említett inkrementális programfejlesztési módszert). A TREE-TYPE modul szerves része az MPROLOG Shellnek, de tetszőleges más MPROLOG programhoz is csatolható.

4.5. MPROLOG Shell: az MPROLOG nyelvet támogató szakértőrendszer-keret

Egy PROLOG program felfogható olyan logikai formalizmussal ábrázolt szabályalapú SZR-nek, melyhez a PROLOG interpreter egy egyszerű hátrafelé haladó stratégiával (és visszalépéses kereséssel) dolgozó következtető gépet biztosít. Az MPROLOG Shell első változatának célja az ilyen szemléletű SZR-ek építéséhez és futtatásához kényelmesen felhasználható szolgáltatások biztosítása. Ehhez felhasználja a fent említett QUERY, DIALOG, valamint TREE-TYPE modulokat.

Az MPROLOG Shell

- a teljes MPROLOG nyelv használatát megengedi (az ismeretbázis építésénél),
- az ismeretbázis elemeit hatékonyan, közvetlenül az MPROLOG interpreter segítségével hajtja meg,

– PC kategóriájú gépekre készül el először, azonban hordozása további 10 géptípusra (munkaállomásokra, nagygépekre) a kompatibilitás magaszintű megtartása mellett az MPROLOG rendszer bázisán viszonylag könnyen megoldható.

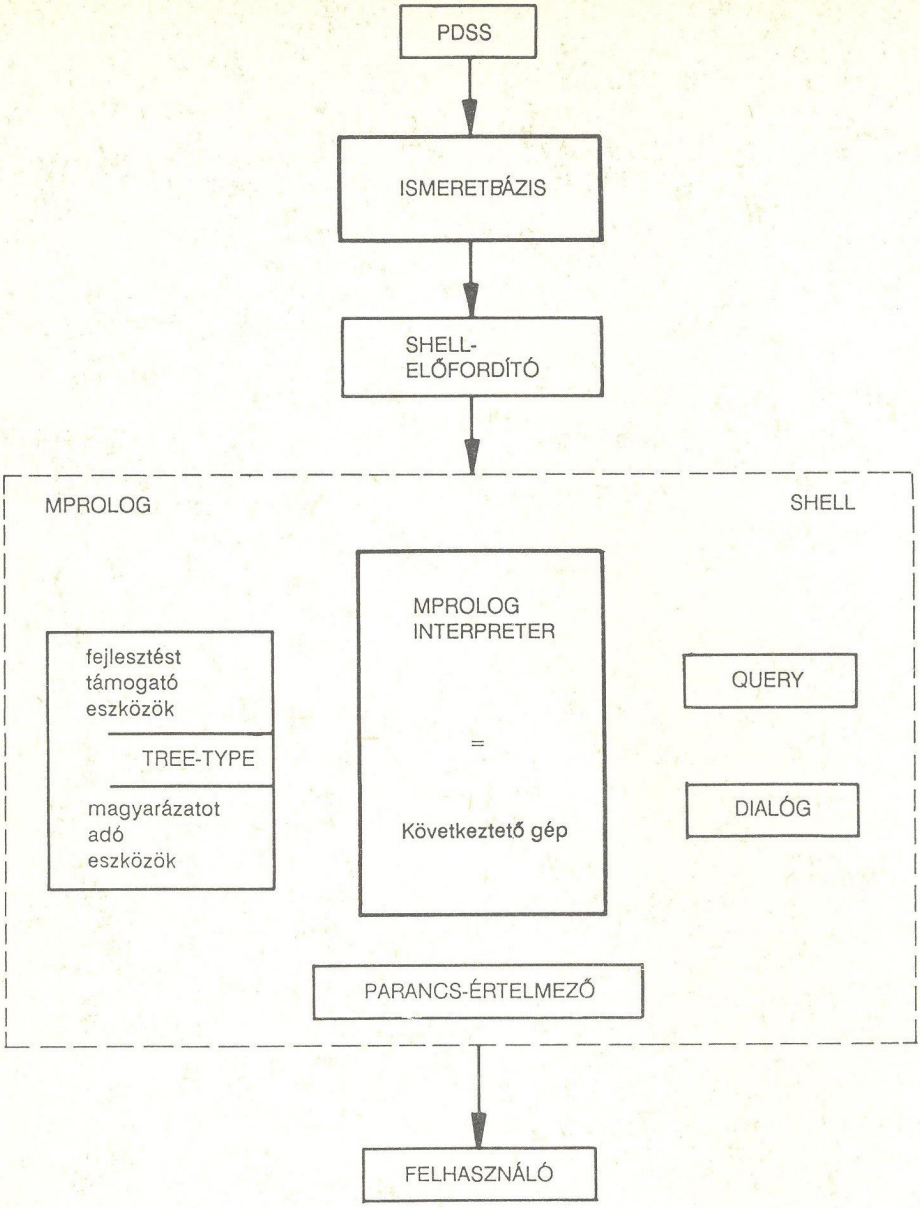
Az MPROLOG Shell moduláris módon továbbfejleszthető az előrehaladó következtetés, bizonytalanságkezelés, frame alapú ismeretreprezentáció, nagyméretű ismeretbázisok kezelése, stb. felé. A 4.2. ábra az MPROLOG Shell első változatának funkcionális felépìtését mutatja (ld. még [MPROLOG SHELL,1988]).

Az MPROLOG alapú SZR-ek ismeretbázisa egy olyan MPROLOG program, amely a tárgyköri ismereteket reprezentáló szabályok mellett a QUERY és TREE-TYPE számára szóló metadeklarációkat is tartalmazza. A fejlesztést éppen ezért célszerű az MPROLOG rendszer programfejlesztő alrendszerének (PDSS: Program Development Subsystem) felhasználásával kezdeni. Az MPROLOG Shellen belül is elérhetők bizonyos, az inkrementális fejlesztést támogató szolgáltatások, azonban azok a rendelkezésre álló tárméret miatt PC-n nem érhetők el.

Az MPROLOG Shell első változata (azonkivül, hogy MPROLOG-ban van írva, tehát az MPROLOG INTERPRETER futtatja,) az MPROLOG INTERPRETER-t következtető gépként is felhasználja, a magyarázatadáshoz szükséges információk kinyerésére az MPROLOG végrehajtási vermet véve segìtségül. Az ismeretbázist tehát közvetlenül az MPROLOG INTERPRETER kezeli, azonban (a magyarázatadás hatékony és igényes megvalósìtása végett) az ismeretbázist egy SHELL-ELŐFORDÍTÓ átdolgozza. A rendszer lehetőséget biztosìt rendszermentésekre, továbbá - a DIALOG rendszerhez hasonlóan - eset-file-ok kezelésére. Az MPROLOG Shell beépìtett "fejlesztő szolgáltatásai" elsősorban a futtatás/tesztelés támogatására szolgálnak - jóllehet, a SZR fejlesztőnek is nagy segìtséget nyújt a magyarázatadás az ismeretbázis tesztelésénél.

Az MPROLOG Shell első változatát először egy nagyméretű ismeretbázissal rendelkező SZR fejlesztésénél alkalmaztuk, amely orvosi témájú: speciális emésztőszervi szakvizsgálatok elvégzésének eldöntésénél nyújt támogatást.

Az MPROLOG alkalmazásokat támogató modulkészlet és keretrendszerek továbbfejlesztése már korábban elkezdődött a fent emlìtett irányokba. Az MPROLOG 2.3 verziójának további (elsősorban micro VAX és Macintosh, majd nagyobb) gépekre való átvitele után a fentiekben ismertetett (és további) modulkészletek és keretrendszerek e gépeken is rendelkezésre fognak állni.



4.2. ábra. Az MPROLOG Shell funkcionális felépítése

5. Hazai szakértőrendszer-keretek összehasonlítása

Az 1. táblázat a hazai fejlesztésű SZRK-ek összehasonlìtását célozza - néhány hazánkban is elérhető, kereskedelmi forgalomban lévő PC-s shell adatait is feltüntetve (forrás: a PC-s SZRKek [Sántáné-Tóth és Sztanev,1987] dolgozatban közölt összehasonlìtó táblázata). Az egyes ismérvek önmagukért beszélnek; részletesebb kifejtésük az idézett helyen megtalálható.

SZRFE összehasonlìtási szempontok	ES/P Advisor	Expert Ease	Insight 2	M.1	Nexpert	ALL -EX	GENE -SYS	MPROLOG Shell
Következtetési módszerek								
*egyszerü	X		X	X		X		X
*struktúrált					X		X	X
*indukciós		X					X	
*logikai	X					X	X	X
Ismeret reprezentációk								
*partíció	E		E	E	T			
T								
*megoldások	T	T	T	T	T	T	T	ET
*bizonytalanság			X	X	X	X	X	
*aritmetika	X		X	X	X	X	X	X
Végrehajtás iránya								
*előrehaladó			X	X	X	X	X	
*hátrafelé haladó	X		X	X	X	X	X	X
*kevert/vegyes					X			
Felhasználói interfész								
*képernyőkezelés	M	M	M	S	M	M	M	M
*bevitel	V	V	V	K	V	V	V	V
*több/bizonytalan válasz	X		X	X	X	X	X	X
*irányított keresés			X		X			
*online help	X		X		X			
*válaszadás sebessége	K	GY	GY	K	GY	K	GY	K

SZRFE összehasonlìtási szempontok	ES/P Advi-sor	Expert Ease	Insight 2	M.1	Nexpert	ALL -EX	GENE -SYS	MPROLOG Shell
Fejlesztői környezet								
*ismeretbázis létrehozása	WP	LE	WP	WP	WP	WP,LE	WP	WP,PDSS
*ismeretbázis módosítása				X	X	X	X	X
*grafika					X			
*HOW & WHY	X		X	X	X	X	X	X
*a következtetés lépéseinek követése	X		X	X	X	X	X	X
*apropó-tallózás				X	X	X	X	X
*tesztesetek kimentése	X		X	X		X	X	X
*formátumozás			X					
Kapcsolat más rendszerekkel								
*beágyazható *adatbázishoz kapcsolat			X	X		X	X	X
*más nyelvü rutinok bekapcs.	P		Pa			P	P,C	P,C,Pa
*munkaállomások, nagy gépek felé kompatibilitás								(X)
Szoftvewr követelmények								
*a megvalósìtás nyelve	P	Pa	Pa	P	A	P	P	P
*lefordìtott változat fut	X	X	X		X	X		(X)
*más nyelvű rutinok					Li			
*operációs rendszer	DOS	UCSD-Pa	DOS	DOS	DOS	DOS	DOS	DOS
*lezárt verzió képezhető	X	X	X	X		X		X
Hardver követelmények								
*gép	P	P,D,V	P,D,V	P,X	M	PX(M)	PX	PX(M)
*tárkapacitás:RAM, kbájt (javasolt)	128	128	192(256)	192	512	640	512(640)	640
*egyéb (javasolt)	(C)		(2D)	(C)		(C)	(10M)(C)	

1. táblázat: Néhány Magyarországon elérhető, valamint három hazánkban fejlesztett SZRK összehasonlító táblázata

Rövidítések:

Partíció:	E = egy	T = több	
megoldások:	E = egy	T = több	
képernyőkezelés:	M = menüvezérelt;	S = soros	
válaszadás sebessége:	K = közepes;	GY = gyors;	L = lassú;

ismeretbázis létrehozása:

WP = szó-processzor; LE = sorszerkesztő

PDSS = Mprolog Program Development Subsystem

nyelvek:	P = Prolog;	C = C;	Pa = Pasal;	Li = Lisp;
gép:	P = IBM PC;	D = DEC;	V = Victor	
	X = IBM/XT;		M:Macintosh	

ALL-EX (1987):
ALL-EX Kézikönyv, 2.1 Verzió.
Alkalmazott Logikai Laboratórium, 1987.

DIALOG (1988):
MPROLOG DIALOG V.2.3.0. Felhasználói kézikönyv.
SZKI, Budapest, 1988.

EIBEN Á.(1986):
Szakértői rendszerek mikrogépen.
Információ Elektronika, 1986/5. 293-302. old.

EXPERT SYSTEM STRATEGIES (1987):
Expert Systems Strategies, Vol.3., No.6, 1987.

FUTÓ I. (1988):
Szakértői rendszerek használata számìtógépes szimulációnál.
in: Szakértő rendszerek'88. Ismeretalapú információfeldolgozás Magyarországon, SZÁMALK, 1988. Ugyanebben a kötetben.

GENESYS (1987):
GENESYS Reference Manual and Tutorial.
SZÁMALK. 1987.

GEVARTER,W.B.(1987):
The nature and evaluation of commercial Expert Building Tools.
Computer, May 1987. pp. 24-41.

GILMORE,J.F. - HOWARD,C.(1986):
Expert Systems Tool Evalution.
Proceedings on the 6th Int. Workshop on ES & Their Application. Avignon (France) April 28-30. 1986. pp. 437-456.

HARMON,P. - KING,D.(1985):
Expert Systems. Artificial Intelligence in Business.
Wiley and Sons Inc., 1985.

HEWETT,J. - TIMMS,S. - D'AUMALE,G.(1986):
Commercial Expert Systems in Europe.
London, 1986. OVUM Ltd.

KOCH P.(1986):
Mikroszámìtógép-alapú szakértői rendszerek ma és holnap.
Információ Elektronika, 1986/5. 293-296.old.

KOCH P.(1987):
A szakértői keretrendszerekről.
Mérés és Automatika, 35. évf., 1987. 12. szám, 433-436.old.

KOCH P.(1987):
SzRlab Koncepció.
SZÁMALK-OMFB, G1-42-100/87, 1987.

LISTEN TO A SELECTION OF EUROPEAN SHELLS.(1987)
Expert Systems Users, Vol.2. 1987. No. 12. p.14.

MOLNÁR K.(1987):
Mprolog logic programming language and the Expert Systems.
To appear in the Proceedings of COMPCONTROL'87.

MPROLOG. (1985):
MPROLOG Documentation.
SZKI, Budapest, September 1985.

MPROLOG SHELL,(1988):
MPROLOG SHELL V.2.3.1. Felhasználói kézikönyv.
SZKI, Budapest, 1988.

PÁSZTOR Z. - SÁNTÁNÉ-TóTH E. (1987):
Számìtógépes szakértő rendszerek alkalmazásának és tervezésének kérdései.
Információ Elektronika, 1987. 4.sz. (Megjelenés alatt.)

REICHGELT,H. - HARMELEN,F. (1987):
Criteria for choosing representation languages and control regimes for Expert Systems.
Univ. of Edinburgh. Manuscript,1986-87.

SÁNDOR G. (1987):
MPROLOG DIALOG - a user interface tool for Expert Systems.
Lecture in the Fifth Symposium on Microcomputer and Microprocessor Applications. Sept-Oct 1987. Budapest.

SÁNTÁNÉ-TóTH E. (1987):
Szakértő rendszer fejlesztések Magyarországon.
Mérés és Automatika 35. évf., 1987. 12. szám, 449-451.old.

SÁNTÁNÉ-TóTH E.,(1988):
Mesterséges Intelligencia - Ismeretalapú Rendszerek.
in: Szakértő rendszerek'88. Ismeretalapú információfeldolgozás Magyarországon, SZÁMALK, 1988. Ugyanebben a kötetben.

SÁNTÁNÉ-TóTH E. - SZTANEV Iné (1986,1987):
A szakértői rendszerek néhány aktuális kérdése az Avignon'86 konferencia és kiállìtás tükrében. I-II.
Információ Elektronika, 1986/6. 307-313.old., 1987/1-2. 3-10.old.

SIMONS,G,L. (1985):
Expert Systems and Micros.
The National Computing Centre Ltd., 1985.
(Magyar nyelvű fordìtása "Szakértői rendszerek és mikrók" cimmel 1987-ben jelent meg a Műszaki Kiadó gondozásában.)

WATERMAN, D.A.(1986):
A Guide to Expert Systems.
Addison-Wesley Publishing Company. 1986