SZÁMITÁSTECHNIKAI KOORDINÁCIÓS INTÉZET

PROLOG fejlesztések és alkalmazások Magyarországon 1975-től 1980-ig

Készült: az SZKCP CF-31 célfeladat keretében,
a SZÁMKI megrendelésére,
az SZKI Elméleti Laboratóriumában
és a NIM IGÜSZI Software-fejlesztési
Osztályán

Készitették: Szeredi Péter
Sántáné-Tóth Edit
Bendl Judit

Budapest, 1980. november

TARTALOMJEGYZÉK

Bevezetés

A PROLOG /PROgramming language based on LOGic/ nagyon magas szintü, leiró jellegü, nem-algoritmikus programozási nyelv. Elméleti alapja a matematikai logika, azon belül az elsőrendü mechanikus tételbizonyitás elmélete. A nyelv nagy előnye az, hogy az elméleti megalapozottság mellett nagyon egyszerü végrehajtási mechanizmussal bir, amely az emberi problémamegoldás köréből vett fogalmakkal könnyen leirható, könnyen megérthető.

A hazai PROLOG-fejlesztések és alkalmazások gyökerei 1973-ra nyulnak vissza. A KSH OSzI /KSH Országos Számitástechnika-alkalmazási Iroda/ támogatásával ugyanis ebben az évben kezdett foglalkozni a NIM IGÜSZI Software-fejlesztési Osztályán néhány kutató a mechanikus tételbizonyitók, valamint a programhelyességbizonyitás és az "ujelvü" nyelvek elméleti és gyakorlati kérdéseivel. Az első eredményekről az [*Andréka,73]*, [*Balogh 75a], [Tóth, 74] áttekintő dolgozatok számolnak be. A lelkes kutatógárda már 1974-ben létrehozta az első magyarországi tételbizonyitó programrendszert [*Balogh 75b] a NIM IGÜSZI ICL 1903A gépén, CDL nyelven /Compiler Description Language, lásd pl. [*Bedő, 79]/. Ez a fejlesztés is a KSH OSzI finanszirozásával jött létre éppugy, mint az első hazai PROLOG-interpreter 1975-ben [Szeredi P, 75a, 75b]. Az ezutáni, a következőkben emlitésre kerülő fejlesztések már a CF-31 célfeladat keretében történtek.

A CF-31 célfeladatot a nagyüzemi software-gyártás és automatizált rendszertervezés és szervezés módszereinek kutatás-fejlesztése és létrehozása érdekében tüzte ki a KSH OSzI. Célja az volt, hogy az 1976-80. közötti időszakban támogassa és finanszirozza a nagyüzemi software gyártás, korszerü, egységes programfejlesztési tech-

---

A *-gal kezdődő hivatkozások a jelen tanulmányban hivatkozott, a Bibliográfiában nem található dolgozatokat felsoroló Irodalomjegyzékre, a jelzetlenek pedig hazai szerzők PROLOG-témáju dolgozatait felsoroló Bibliográfiára utalnak

nológia kialakitása hazai feltételeinek megteremtését. A "korszerü" jelző megkivánta a külföldi - már alkalmazott, ill. még csak kifejlesztés alatt lévő - software előállitást célzó technológiák tanulmányozását, elemzését, ennek alapján a hazai teendők meghatározását és elvégzését. Ennek a munkának szerves része volt a jelen beszámoló tárgyát képező PROLOG, tágabb értelemben véve a logikai alapu programozási technológia eszközeinek meghonositása, továbbfejlesztése, gyakorlatba vitele. E munkát a NIM IGÜSZI fejlesztő kollektivája végezte; 1978-tól az SZKI Elméleti Laboratórium munkatársai is bekapcsolódtak a fejlesztésbe. 1979-től kezdve a PROLOG terjesztésével, kisérleti, valamint gyakorlati alkalmazásba vételével további intézmények kapcsolódtak bele a munkába.

A kutatási, fejlesztési eredményekről készitett beszámolók, tanulmányok a /CF-31 célfeladat koordinálásával megbizott/ SZÁMKI által gondozott SOFTTECH sorozatban jelentek meg. Természetesen számos cikk jelent meg, évente több előadás hangzott el nemzetközi és hazai rendezvényeken, és sok intézeti belső anyag került kidolgozásra és terjesztésre ezzel kapcsolatban. Ezen dolgozatok jegyzékét jelen összeállitás Bibliográfiája tartalmazza; ez a megjelent publikációkat /beleértve ezek közé a SOFTTECH-sorozatban megjelent tanulmányokat, dokumentumokat is/ mind tartalmazza; a felsorolásba bevettük még az intézeti belső felhasználásra készitett fontosabb anyagokat is.

Az alábbiakban röviden áttekintjük a PROLOG-fejlesztések és -alkalmazások eddigi fontosabb állomásait.

1975 közepére készült el - a [xBattani, 73]-ban közzétett nyelv-specifikációt és interpreter-megvalósitást tartalmazó anyag alapján - az első hazai PROLOG interpreter a NIM IGÜSZI ICL 1903A gépére, CDL nyelven. /A korábban, 1972-ben ben Marseille-ben IBM 360-ra készitett PROLOG interpreter FORTRAN nyelven készült. / Ebben az évben már müködtek az első, az épitészeti tervezés köréből, valamint a gyógyszervegyészet területéről származó PROLOG-alkalmazások /3.3.1, ill. 3.1.4/. Még ez évben installálásra került az interpreter az ICL SYSTEM 4/70 gépén; itt a gép konfigurációja és a MULTIJOB operációs rendszer nagyobb lehetőségeket kinált a felhasználóknak és fejlesztőknek egyaránt.

1976-ra az ICL 1903A gépen, batch környezetben müködő interpreter programbelövési segédletekkel, a felhasználó kényelmét szolgáló további szolgáltatásokkal bővült. A gyógyszervegyészeti, valamint az épitészeti tervezés területéről további feladatok kerültek megoldásra /3.1.3, 3.2.1 és 3.3.2/; megjelentek az első, PROLOG-alkalmazással kapcsolatos publikációk. Ebben az évben további három gépen, az EMG 840, a HWB 66/40 és az ODRA 1304 gépeken került installálásra az interpreter.

A NIM IGÜSZI munkatársai az alábbi munkákkal párhuzamosan foglalkoztak logikai alapu kérdés-válasz rendszerek vizsgálatával is /Horváth,76/.

1977-ben befejeződött az interpreter hangolása, aminek következményeképpen négyszeresre gyorsult fel a PROLOG programok futása. Ez a tény erős mértékben hozzájárult a komoly, "éles" gyakorlati alkalmazások megjelenéséhez. Továbbfejlesztésre kerültek a korábbi gyógyszervegyészeti alkalmazások /3.1.3, 3.1.4, 3.2.1/, és megjelent az első két információ-visszake-

reséssel foglalkozó alkalmazás: ezek a levegőtisztaságvédelem, valamint a növényi kártevők és növényvédőszerek kérdéseivel foglalkoztak /3.2.2 és 3.2.3/. Ebben az évben erőteljesen megindult a PROLOG terjesztése: installációk jelentek meg az ICL 1905, a két ODRA 1305, továbbá a SIEMENS 7.740 /később 7.755/ gépeken. A SIEMENS család több lehetőséget biztositó, 7.755 számu tagjának üzembeállitása után megindult az interpreter interaktiv szolgáltatásokkal rendelkező változatának kifejlesztése. Még ez évben installálásra került az interpreter az IBM 370/145 gépen is /DOS/VS alatt/.

1978-tól a PROLOG-kutatásokban fordulat állt be. Ebben az évben - az eddigi PROLOG-fejlesztések betetőzéseként - befejeződött a PROLOG interaktiv változatának kidolgozása. Az SZKI SIEMENS 7.755 gépén, BS2000 alatt müködő interpreter azzal, hogy interaktiv programfejlesztési környezetet, kényelmes szimbólum-manipulációs szolgáltatásokat biztositott, továbbá hogy az operációs rendszer virtuális tárkezelése következtében az eddigieknél jóval nagyobb felhasználói tárat tudott biztositani, visszahatott az alkalmazásokra: mélyültek az addigi alkalmazások, további /eddig reménytelennek tünő/ területekről jelentkeztek feldolgozási igényel. Ez az interpreter képezte bázisát mind a további, ESzR-kompatibilis PROLOG-installálásoknak, mind pedig az ezután meginduló, PROLOG-alapu rendszer-fejlesztéseknek. Ebben az évben került installálásra az interpreter OS-változata az R22 és az IBM 370/145 gépekre. Megjegyezzük, hogy ugyanez az interpreter a későbbi években installálásra került egyes szocialista országokban is /Kiev: 1979, Prága: 1980/.

A fontosabb uj alkalmazások közül ki kell emelnünk a COBOL nyelvü programok generálását /3.4.1/, számitógép-architektura tervezését /3.5.1/, és mikroprogramok beültetését /3.4.3/. Két további uj terület olyan feladatokat kinált, amelyek ujabb kutatási irányokat jelöltek meg: szimuláció /3.6.1 korai változata/ és programtervezés /3.4.6/. Mindkét irányban - mint látni fogjuk - önálló PROLOG-alapu rendszerek jöttek létre a későbbiekben: a T-PROLOG és az LDM.

1978-ban indult be hazánkban az a munka, amely korszerü lehetőségek /pl. modul-kezelés/, hatékonyabb eszközök /pl. előforditó-menet, majd forditóprogram/ biztositására, ill. kifejlesztésére irányult. A fejlesztést alapos irodalmazás előzte meg: a külföldről érkező, PROLOG-fejlesztésekről szóló anyagok közül jónéhány foglalkozott ugyanis a PROLOG-programok futásidejének és helyigényének, PROLOG forditóprogram előállitásának részleteivel. A hazai elképzelések alapján 1978-ban készült el az MPROLOG nyelv /azaz a moduláris PROLOG nyelv/ specifikációja [Bendl,78].

1979-ben a hazai PROLOG-alkalmazók és alkalmazási feladatkörök számának növelése érdekében a KSH OSzI a CF-31 célfeladat keretében anyagi lehetőségeket biztositott. Célja az volt, hogy az arra igényt tartó intézményeknek lehetőséget nyujtson arra, hogy jelentősebb anyagi kockázatvállalás nélkül készülhessenek fel a PROLOG nyelvnek saját feladatkörükön belül történő felhasználására. A célfeladatot koordináló SzÁMKI az SZKI-t és a NIM IGÜSZI-t bizta meg uj PROLOG-alkalmazások felkutatásával, és azok támogatásával. E két intézmény ennek során

- felkutatott olyan intézményeket, ahol PROLOG-gal hatékonyan megoldható alkalmazói feladatok megoldása időszerü volt,

- konzultációs segitséget nyujtott a feladatok megfogalmazásához és a PROLOG nyelvü programok megirásához, továbbá

- segitséget nyujtott a PROLOG rendszernek a felhasználó gépén történő üzembehelyezéséhez.

A következő hazai intézmények támogatására került sor: Általános Géptervező Iroda, Épitéstudományi Intézet /ÉTI/, Ho Si Minh Tanárképző Főiskola Budapesti Tagozata, KFKI MSZKI Számitástechnikai Főosztály, KSH Nemzetközi Számitástechnikai Oktató- és Tájékoztató Központ /SZÁMOK/, MTA Nyelvtudományi Intézet, MTA SZBK Enzimológiai Intézete, NIM IGÜSZI Kőolaj- és Gázipari Rendszerszervezési Osztály, Pollack Mihály Müszaki Főiskola /PMMF/ Matematika és Számitástechnikai Intézet.

A hazai intézmények támogatása mellett tovább folytatódtak a korábban beindult fejlesztések: az MPROLOG, a T-PROLOG, az LDM.

Ebben az évben a SIEMENS 4004, az R22/DOS/ és az R40 gépekre történt meg a PROLOG interpreter installálása. Ezen kivül a következő területeken jöttek létre kisérleti, ill. éles alkalmazások: gyógyszervegyészeti tervezés /3.1.5, 3.1.6, 3.1.7/, épitészeti tervezés /3.3.4/, software alkalmazások /3.4.2, 3.4.4/, nyelvészet /3.7.1/, egyéb alkalmazások /3.7.2, 3.7.3/. Ebben az évben müvelt további alkalmazások: gyógyszervegyészet /3.1.1, 3.1.2, 3.1.4/, információ-visszakeresés /3.2.4/, COBOL program generálás /3.4.2/, architektura-tervezés /3.5.2/, szimuláció /3.6.2/.

1980 a SIEMENS 7.755 operációs rendszer környezetében kivitelezett /befejeződő és ujként meginditott/ fejlesztések, valamint további installálások és sikeres alkalmazások éve volt.

Az eddig bázisként felhasznált PROLOG-interpreter fejlesztését az MPROLOG-rendszer interpreterének hangolása, kényelmes programfejlesztési környezet biztositása, gyorsitása, valamint a gyakorlati felhasználás közben jelentkező hibák kijavitása váltotta fel. Elkezdődtek az MPROLOG rendszerhez kapcsolódóan a forditóprogramos változat tervezési munkái is. Befejeződött a T-PROLOG rendszer kifejlesztése, és elkezdődött /az MPROLOG-rendszer felett/ az LDM első, kisérleti változatának realizálása. E két utóbbi, PROLOG-alapu rendszerről szól a 4.1-4.2 pont./ Itt kivánjuk megemliteni, hogy 1979-80-ban az LDM-kutatásokat - ESzR-MSzR TKM 3. fejezetében foglalt egyik részfeladatként - az OMFB finanszirozta. A T-PROLOG az SzKI belső fejlesztéseként jött létre./

A PROLOG bázisrendszer ebben az évben installálásra került az IBM 3031 gépre. Az MPROLOG interpreter is terjesztésre került: a SIEMENS 7.755 gépen kivül müködik már az IBM 3031 és az R22/OS/ gépeken is. Befejeződéshez közelednek az MPROLOG első alkalmazásai is; ezek az LDM kisérleti rendszer, valamint a 3.2.4-ben leirt információs rendszer. További eredményesen müvelt PROLOG alkalmazási területek: épitészeti tervezés /3.3.3/, program-analizis /3.4.4/, programdokumentáló rendszer /3.4.6/, szimulációs modell generálás /3.6.2/. A gyógyszervegyészeti alkalmazások rendszeresen visszatérő PROLOG-futtatásokkal, továbbfejlesztési elképzelésekkel jelentkeztek. Az alkalmazási igények azonban 1980-ra már kinőtték a PROLOG bázis-implementációt. Pillanatnyilag a legégetőbb kérdés az alkalmazási programok futásidejének optimalizálása.

Visszatekintve a hazai PROLOG-fejlesztések eddigi 6 évére, külön ki kell emelnünk, hogy a PROLOG interpreter CDL, az

MPROLOG interpreter pedig CDL2 nyelven készült. Az implementálás nyelvének ezen megválasztása nagyban megkönnyitette mind a fejlesztést, mind pedig a terjesztést; nem véletlen, hogy az ANSWER rendszer jelenlegi változata is éppen CDL2 nyelvü programok kifejlesztését támogatja. Érdekes - és ez sem véletlen - hogy az ANSWER rendszer két komponense, az LDM tervezési és az információs alrendszer első változatának implementációs nyelve az MPROLOG.

A hazai PROLOG fejlesztés és alkalmazás elismerését mutatja, hogy az 1980 évi "Logikai Programozás Szeminárium" megtartására a nemzetközi programbizottság a Neumann János Számitógéptudományi Társaságot kérte fel. Az 1980 nyarán Debrecenben megrendezett szeminárium kiemelkedően nagy érdeklődés mellett, hazai PROLOG-fejlesztők és alkalmazók, valamint neves külföldi PROLOG-fejlesztők részvételével zajlott le [LPW,80]. A magyar résztvevők köréből hat előadás hangzott el hazai fejlesztésekről és alkalmazásokról. Ez a seregszemle jól bizonyitotta, hogy mind a hazai PROLOG-fejlesztések, mind pedig az alkalmazások nemzetközi mércével mérve igen jó eredményeket értek el az elmult években.

Itt köszönjük meg a jelen anyagban emlitett minden PROLOG-fejlesztő és -alkalmazó segitségét, amellyel hozzájárultak ezen összeállitás elkészitéséhez. Végezetül hadd idézzük az ÉTI és az SzKI által közösen kiadott, a PROLOG kézikönyvet [Szeredi P,77a] és a PROLOG bázisrendszer uj szolgáltatásait [Köves,78] tartalmazó "PROLOG" cimü kötetének előszavából:

"Amikor megköszönjük a KSH OSzI CF-31 keretében nyujtott önzetlen segitségét, közösen azt kivánjuk minden alkalmazónak, hogy mielőbb megérezze e nem algoritmikus nyelv lehetőségeit, s azt szakterületén sikerrel kamatoztassa."

Az anyag további részében a PROLOG /ill. MPROLOG/ nyelvet, majd interpretereiket mutatjuk be /1. és 2. fejezet/. Ez után témánként csoportositva felsoroljuk az elmult időszak lényegesebb PROLOG-alkalmazásait /3. fejezet/, vázoljuk a további fejlesztési elképzeléseket /4. fejezet/, majd összefoglalásként rövid értékelését adjuk az eddigi hazai PROLOG-fejlesztéseknek és alkalmazásoknak /5. fejezet/. Az anyaghoz Irodalomjegyzék és Bibliográfia van csatolva /az Irodalomjegyzékre való hivatkozásokat * jellel különböztetjük meg/. A Bibliográfia a hazai szerzők PROLOG-témáju dolgozatait a teljesség igényével tartalmazza, mig az Irodalomjegyzék a jelen beszámolóban hivatkozott anyagokhoz készült. Utóbbi csak a Bibliográfiával együtt teljes; nem vettük fel ugyanis az Irodalomjegyzékbe azokat a hivatkozásokat, amelyek a Bibliográfiában megtalálhatók.

1. A PROLOG nyelv bemutatása

A számitástechnika hardware és software eszközeinek fejlődésével egyre bonyolultabb feladatok számitógépes megoldása válik lehetővé. Ebből adódik, hogy egyre több szakterület feladatai válnak megoldhatóvá a számitógép segitségével. E feladatok megoldása során egyre több nem-számitógépes szakember kerül többé-kevésbé közvetlen kapcsolatba a számitógéppel. Ez a tény indokolja azokat a kutató-fejlesztő munkákat, amelyek célja az ember és a számitógépes rendszerek közötti kapcsolatteremtés és -tartás megkönnyitése, emberközelibbé tevése. A könnyités egyik módja az egyes szakterületek feladataira orientált célrendszerek kifejlesztése, másik módja pedig a programfejlesztési munkákat könnyitő eszközök kidolgozása. Mivel a számitógépes rendszerek általános alkalmazhatóságát ez utóbbi megoldások őrzik meg, a kutató-fejlesztő munkák előterében a programfejlesztést könnyitő eszközök - a programozási rendszerek - állnak.

Az ilyen irányu kutatások eredményeinek egyik csoportját azon programozási rendszerek alkotják, amelyek a hagyományos értelemben vett számitógépes programok kidolgozását segitik a kidolgozás egy vagy több, esetleg minden fázisának támogatásával /pl.: ISDOS, ANSWER, SOFTING stb./. E programozási rendszerek közös vonása, hogy a felhasználásukkal készült végtermék - a feladat megoldására szolgáló program - végülis egy hagyományos algoritmikus programozási nyelven áll elő: a program valamilyen nyelven a feladat megoldásának módját rögziti.

A programozási rendszerekkel kapcsolatos kutatások másik iránya a programozási nyelv szintjének emelését; a leiró nemalgoritmikus, un. ujelvü programozás megvalósitását tüzte ki

célul. Az irányzat fő jelszava a "MIT és nem HOGYAN". E jelszó arra a célkitüzésre utal, hogy a számitógép felhasználójának lehetőleg csak magát a feladatot kelljen megfogalmaznia, tehát azt, hogy a számitógéppel mit kiván megoldani, és mindazzal, hogy ezt a gép hogyan oldja meg, ne kelljen foglalkoznia.

Az ujelvü programozási nyelvek egy része egy hagyományos algoritmikus nyelvet bővit olyan magasszintü elemekkel, amelyek a leiró programozást támogatják. Ennek a közelitésmódnak nagy előnye, hogy a felhasználóknak viszonylag kevés uj programelemet kell megismerniük, megtarthatják "hagyományos" programozási módszereiket, szokásaikat és csak szükség esetén kell alkalmazniuk az uj elemeket.

Az ujelvü programozási nyelvek másik csoportja matematikai alapokra épül. Ennek az irányzatnak legjobban kidolgozott területe az un. "logikai programozás" /logic programming/. A logikai programozás alapelve az, hogy magukat a logikai állitásokat is értelmezhetjük programként, ha egy megfelelő végrehajtási mechanizmussal látjuk el a logika nyelvét. Ez azzal a nagy előnnyel jár, hogy a "logika"-programok értelmezhetők leiró módon, logikai állitásként /MIT-rész/, de ugyanakkor tartozik hozzájuk egy végrehajtási mechanizmus is /HOGYAN-rész/. A logikai programozás elvének megvalósitásában a legnagyobb nehézséget a megfelelő kompromisszum kialakitása okozza: ha a logika szintje tul magas, nem található hozzá elég hatékony végrehajtási mechanizmus, a logika szintjének tulságos csökkentésével viszont a nyelv veszit kifejező erejéből, a leiró jelleg akár el is tünhet.

A PROLOG nyelv /PROgramming language based on LOGic - logikai alapu programozási nyelv/ a mai hardware és software körülmé-

nyek között is sikeresen oldja meg a logikai programozás megvalósitási problémáit. A PROLOG nyelv az elsőrendü logika nyelvének egy megszoritása /az un. Horn-logika/, végrehajtási mechanizmusa pedig nem más, mint mintaillesztéses paraméterátadással és visszalépéses kereséssel ellátott rekurziv eljárásszervezés. A PROLOG nyelv kifejezőereje /a megszoritások ellenére is/ jelentősen tullépi a mai magasszintü nyelvekét; jogosan nevezhető tehát "nagyon magas szintü" nyelvnek.

1.1 A PROLOG nyelv alapelemei

A PROLOG nyelv nagyon egyszerü, mégis nagy kifejezőerővel rendelkező programozási nyelv. A PROLOG programok elemei logikai állitások, amelyek valamely reláció /összefüggés/ fennállását fejezik ki. A reláció fennállása bizonyos feltételek teljesüléséhez is köthető, a PROLOG-ban leirható logikai állitások alakja tehát

$\langle \text{reláció}_0 \rangle$ igaz akkor, ha

$\langle \text{reláció}_1 \rangle$ és ... és $\langle \text{reláció}_n \rangle$ is igaz.

azaz az állitás a következmény /$\langle \text{reláció}_0 \rangle$/ fennállását a feltételben szereplő relációk együttes fennállásához köti.

A relációk argumentumaiként egyszerü vagy összetett kifejezések szerepelhetnek. Egyszerü kifejezésként változók, számok, vagy tetszőleges nevek /azonositók/ szerepelhetnek. Összetett kifejezésként tetszőleges számu mezőből felépitett, névvel ellátott összetett adatstrukturát /rekordot/ használhatunk, ahol a mezők szintén egyszerü, vagy összetett kifejezések lehetnek:

$\langle \text{rekordnév} \rangle (\langle \text{részkifejezés}_1 \rangle , \ldots , \langle \text{részkifejezés}_n \rangle .$

Ez lehetővé teszi a hagyományos programozási nyelvek különféle adatstrukturáinak megfelelő PROLOG kifejezések képzését. Az összetett kifejezések olvashatóbb, szemléletesebb alakban való irását teszi lehetővé az infix alak, amikor egy két részből álló rekordstruktura nevét

/amely egyetlen jel, pl.: +, -, stb. is lehet/ a két rész--kifejezés közé irjuk:

<rész-kifejezés$_1$> <rekordnév> <rész-kifejezés$_2$>

Igy az egyes alkalmazási területek megszokott irásmódjához közelálló kifejezések képzésére nyilik lehetőség, pl.:

"H.2-O" /a H_2O molekula jelölésére/
"LABEL: LOAD. A+2" /egy assembly utasitás jelölésére/.

/Itt a "-", ".", ":", "+" jelek szolgálnak rekordnévként/.

Különleges szerepet játszik a PROLOG-ban a változó fogalma. Ha egy állitásban szerepeltetünk egy változót, ez azt jelenti, hogy az az állitás a változó tetszőleges értéke mellett alkalmazható - ez megengedi, hogy relációkat ne az egyes konkrét összefüggések felsorolásával irjunk le, hanem általános törvényszerüségek megadásával vezessük vissza őket más relációkra.

Mindeddig leiró szemléletben tekintettük át a PROLOG elemeit. Ugyanezek az elemek értelmezhetők procedurálisan is, a hozzájuk tartozó végrehajtási mechanizmusok oldaláról. Ez utóbbi szemléletben egy PROLOG állitás egy eljárásdefiníciónak felel meg: az állitás következményrészében szereplő reláció az eljárás feje, a feltétel részben szereplő relációk pedig az eljárástörzset alkotó eljáráshivó utasitások. Az eljárások logikai eredményt adnak a reláció teljesülésének megfelelően: egy eljáráshivás igaz eredményt ad, azaz sikerül, ha a reláció teljesül, egyébként, vagyis ha nem teljesül, hamis eredményt ad, azaz sikertelenül fut le. Az eljárásba való belépés feltétele az, hogy a hivásban és az eljárás fejében szereplő argumentum-kifejezések összeilleszthetők, azaz azonos alakra

hozhatók legyenek /ehhez a bennük szereplő változókba tetszőleges kifejezések behelyettesitése van megengedve/. Az összeillesztéshez szükséges változó-behelyettesitéseket a hivott és hivó eljárások törzsében is elvégezzük /paraméterátadás/, majd belépünk az eljárás törzsébe /az állitás feltétel-részébe/ és az ott szereplő eljáráshivásokat sorra végrehajtjuk. Miután egy eljárásnévhez /relációhoz/ több definició /állitás/ is tartozhat, ezek alternativ kipróbálására visszalépéses keresés történik: ha valamely ponton a végrehajtás sikertelen ágra jut /egy hivás nem illeszthető egyetlen szóba jöhető definicióval sem/, akkor a legutolsó választási pontra visszatérve egy uj alternativa választásával folytatódik a keresés.

Mint a korábbiakban láttuk, a be- és kimenő paraméterek kezelése egységesen a mintaillesztés segitségével történik. Igy hivásról hivásra változtathatjuk azt, hogy éppen melyik argumentumpoziciót használjuk bemenő, és melyiket kimenő argumentumként.

A végrehajtási mechanizmus oldaláról szemlélve a PROLOG változó-fogalmát, az gyökeresen eltér a hagyományos nyelvek változó fogalmától. A PROLOG-változó kétféle állapotban lehet: értékkel biró és értékkel nem biró állapotban. A változók létrehozásuk pillanatában értékkel nem biró állapotban vannak és az eljáráshivások során, az illesztések folyamán kaphatnak értéket /"helyettesitődhetnek be"/. Ha a változó értékkel bir, uj értéket nem kaphat /nincs értékadó utasitás/, azonban ha visszalépés során az ő behelyettesitését kiváltó eljárás választását megmásitjuk, akkor a változó ujra behelyettesitetlen

állapotba kerül. Ugyanakkor a változók behelyettesitetlen állapotban is előfordulhatnak összetett kifejezések belsejében, azaz az eljárások "félig előállitott" kifejezéseket is kezelhetnek, ami hagyományos programozási nyelvben csak pointerek /referenciák/ képzésével valósitható meg.

Érdekességként megemlitjük, hogy egy PROLOG program probléma-redukciós szemléletben is vizsgálható; ez esetben a program probléma-megoldási /probléma-redukciós/ sémákat tartalmaz. A programfutás során célként kitüzünk egy megoldandó problémát, és az interpreter a programban megadott probléma-megoldási sémák segitségével megoldja az adott problémát /ha az egyáltalán lehetséges/.

1.2 Ujelvü programozás PROLOG-ban

Az a tény, hogy a PROLOG-ban tisztán logikai nyelven programozhatunk, minőségi ugrást jelent a nagyon magas szintü nyelvek célja - a programozás automatizáltabb szintre emelése - irányában. Erre az összefoglaló irásban ezt a minőségi előrelépést egy példa kapcsán szeretnénk megvilágitani. A PROLOG részletes ismertetése iránt érdeklődőknek a [Szeredi P,77a] kézikönyvet ajánljuk.

Példaként vegyük az egyszerü láncolt lista fogalmát. Az ilyen listák ábrázolására PROLOG-ban a "." összekötő jelet használhatjuk, pl. 4.1.2.NIL a 4,1,2 elemekből felépitett listát jelöli. A mintaillesztési mechanizmusnak megfelelően *X . *L egy olyan listát jelölhet,amelynek első eleme /a feje/ *X, és a fennmaradó része /a farka/ *L, /PROLOG-ban minden változó elé * jelet kell irni./ A listákkal kapcsolatos alapvető PROLOG eljárás az alábbi:

```
+ELEME (*X , *Y.*L) - EQUAL (*X , *Y).
+ELEME (*X , *Y.*L) - ELEME (*X , *L).
```

/Itt + jel előzi meg az eljárásfejet, a - jel pedig az eljárástörzsben az egyes eljáráshivó utasitásokat. Egy eljárás több alternativ definicióból állhat, minden eljáráshivás logikai eredményt szolgáltat./

A fenti eljárásdefiniciót pl. az alábbi hivással aktivizálhatjuk:

```
-ELEME (1, 4.1.2.NIL)
```

Ilyen jellegü hivásra a fenti definició kereső eljárásként

müködik, azaz keresi a listában a megadott elemeket, és akkor ad igaz eredményt, ha megtalálta. Ezt az értelmezést az alábbi Algol-szerü programmal irhatjuk le:

```
Boolean procedure eleme ( x, lista);

      if lista ≠ nil then
        begin y:= feje (lista);
              if x = y then eleme : = true
                       else eleme : = eleme ( x, farka (lista))
                    end else eleme := fail;
```

A korábbi PROLOG alak azonban ennél sokkal többet fejez ki; az "ELEMÉ-nek lenni" reláció logikai leirását tartalmazza:

*X eleme egy *Y . *L listának, ha *X = *Y.
*X eleme egy *Y . *L listának, ha *X eleme *L-nek.

A PROLOG mintaillesztésen és visszalépésen alapuló vezérlési mechanizmusa képes ezt a definiciót teljes egészében kihasználni, azaz képes felelni minden olyan kérdésre, amelyre a válasz az ELEME logikai definiciójából kikövetkeztethető.

Ha pl. egy

-ELEME (*X, 4.1.2.NIL)

hivást irunk le, ahol *X értékkel még nem biró változó /jelentése: mondj egy *X-et, amely ELEME a megadott listának/, akkor a hivás hatására *X értéke a második paraméterben megadott lista első eleme, azaz 4 lesz. Ha a fenti hivás után *X-re egy ujabb feltételt is megadunk, pl.

```
-ELEME (*X, 4.1.2.NIL)  -LESS (*X, 2)
```

akkor a visszalépési mechanizmus segitségével a PROLOG *X-ben előállitja a lista első olyan elemét, amely a további feltételnek megfelel /azaz a fenti hivássorozat egy olyan *X elemet ad, melyre *X<2; ez *X = 1 lesz/. Egy másik ilyen jellegü alkalmazást mutat az alábbi példa:

```
-ELEME (*X, 4.1.2.NIL)  -ELEME (*X, 3.2.5.NIL)
```

amelynek lefutása után *X értéke a két lista közös eleme, a 2 lesz.

A fenti ELEME-definició egyik legérdekesebb hivási módja az, amikor a második paraméterben /a lista helyén/ szerepeltetünk változót, pl.:

```
-ELEME (4, *L)  -ELEME (1, *L)  -ELEME (4, *L)
```

Ennek a hivássorozatnak a hatására *L értéke 4.1.*L2 lesz, ahol *L2 egy uj /értékkel még nem biró/ változó. Egy ilyen 4.1.*L2 "részlegesen kitöltött" adat az összes olyan listát jelöli, amely a 4, 1 elemekkel kezdődik. Az

```
-ELEME (a, *L)
```

alaku hivások hatása, ahol *L ilyen részlegesen kitöltött lista - ismét az általános mintaillesztési mechanizmus hatására - a következő lesz: ha *L-ben szerepel már az "a" elem, akkor *L nem változik, egyébként pedig az *L lista a végén bővül egy láncszemmel, amely az "a" elemet tartalmazza /azaz "bekerül" "a"-t a listába/.

Egy ilyen betevő eljárás hagyományos nyelven való leirása már meglehetősen komoly eszközöket igényel:

- uj láncszemek létrehozását /generálását/ és az ezzel kapcsolatos memóriagazdálkodást,
- pointerek /referenciák/ kezelését,
- adatstrukturák mezőinek kiválasztását /erre a korábbi példáinkban is szükség volt/.

Mindezeket a magasszintü eszközöket a PROLOG változófoglalma és mintaillesztési mechanizmusa automatikusan biztositja, mégpedig a tömör, leiró jellegü definició alapján.

Befejezésül bemutatjuk az "ELEME" eljárás definicióját az MPROLOG szintaktikus konvenciói szerint:

```
eleme (X, Y.L)
     when
        X = Y  or     eleme (X, L).
```

Mint látható, ez a szintaxis-alak általánosabb és a logikai jelölésrendszerhez, a leiró-szemlélethez közelebb áll. Ugyanakkor MPROLOG-ban megengedett a logikai összekötő jelek /when, and, or/ helyett a CDL-ben használatos : , ; összekötő jelek használata is, amivel CDL-szerüen irhatók le az MPROLOG programok, pl.:

```
eleme (X, Y.L):
          X = Y;
          eleme (X, L).
```

1.3 A PROLOG nyelv jellemzőinek összefoglalása

Az alábbiakban tömören összefoglaljuk a PROLOG nyelv legfontosabb jellemzőit:

/1/ A nyelv egyaránt rendelkezik egy deklarativ, leiró és egy procedurális, algoritmikus szemantikával; a leiró szemantika megegyezik a matematikai logika természetes szemantikájával.

/2/ A nyelv végrehajtási mechanizmusának elemei:

- /rekurziv/ eljárásszervezés;
- mintaillesztéses paraméterátvétel;
- visszalépéses keresés.

/3/ A nyelv által biztositott adatstrukturák:

- általános rekordstrukturák, amelyek tetszőleges számu almezőből állhatnak, tetszőleges mélységben egymásba-skatulyázva; a rekord felépitès és mezőkiválasztás müveletei mintaillesztéssel valósithatók meg;
- PROLOG-változó, amely az értékadási és pointerkezelési lehetőségeket automatikusan biztositja.

/4/ A nyelv "nagyon magas szintü" lehetőségei:

- többcélu eljárások; egyetlen definició a paraméterek bemenő-kimenő szerepének változtatásával használható az egy relációra vonatkozó összes lehetséges eljárás megvalósitására;
- egyetlen eljáráshivással - a visszalépés segitségével - rendre előállithatók a feladat alternativ megoldásai; ez a magasszintü iteráció egy formája;

- egy eljárás szolgáltathat részlegesen kitöltött /szabad változókat tartalmazó/ eredményeket, amelyeket más eljárások tehetnek teljessé.

/5/ A nyelv kifejezőerejére, használhatóságára, tanithatóságára vonatkozó jellemzők:

- a PROLOG szimbolikus nyelv: relációk, objektumok, adatstrukturák elnevezésére tetszőleges nevek használhatók;
- az adatstrukturák infix alakja a természetes- és szaknyelvekhez közelálló jelölés-rendszert biztosit;
- a számitástechnikai ismeretekkel nem rendelkező, csak az alkalmazási területet ismerő szakember a PROLOG-programok olvasását /a nyelv leiró jellegü értelmezése alapján/ szinte azonnal megtanulja, és könnyen irhat egyszerübb PROLOG programokat.

/6/ A nyelv kapcsolata a "hagyományos" számitástechnikai környezettel az un. beépitett /szabványos/ eljárások segitségével valósul meg. A beépitett eljárásokon keresztül használhatók a gép aritmetikai müveletei, az input-output lehetőségek. Különleges beépitett eljárásokkal befolyásolható a PROLOG vezérlési /keresési/ mechanizmusa, illetve módositható a reláció definiciókból képzett adatbázis.

/7/ A PROLOG nyelv - eltekintve az implementációnként esetleg eltérő, szintaktikus jelölésrendszertől és a felhasználó kényelmét biztositó szolgáltatásoktól - teljes mértékben gép- és implementáció-független.

2. A PROLOG megvalósitásai

A PROLOG programozási nyelvet A.Colmerauer, R.Kowalski és P.Hayes munkássága nyomán dolgozták ki Marseille-ben 1972-ben [*Colmerauer,72]. Az első PROLOG implementáció ALGOL-W nyelven készült, ezt később FORTRAN-IV-be is átirták. Ez az implementáció IBM 360 gépen futott.

Az utóbbi FORTRAN-nyelvü PROLOG interpreter 1973-tól kezdődően számos kutatóhelyen került installálásra. Ennek kapcsán azonban hamarosan fény derült az implementáció hiányosságaira, és megindult az ujabb PROLOG implementációk kifejlesztése.

Az 1975-ben Magyarországon elkészült PROLOG-interpreter [Szeredi P, 75a, 75b] az első volt az ujabb PROLOG implementációk sorában. Az 1975-79 években számos további külföldi PROLOG implementáció született, amelyek közül kiemelendő az Edinburgh-ban és Lisszabonban megvalósitott PROLOG-forditó [*Warren,77] /a lényegesen hatékonyabb megvalósitás miatt/, valamint a londoni Imperial College-ban készült IC-PROLOG [*Clark 80] /a lényegesen uj vezérlési eszközök miatt/.

A PROLOG hazai alkalmazási tapasztalatai, valamint az emlitett uj implementációk tanulmányozása alapján került sor 1979-80 során egy uj hazai PROLOG implementáció, az MPROLOG elkészitésére. Ennek egyik fő ujdonsága a modularitás biztositása /a MPROLOG név kezdőbetüje a modularitásra utal/.

2.1 A PROLOG első hazai megvalósitása

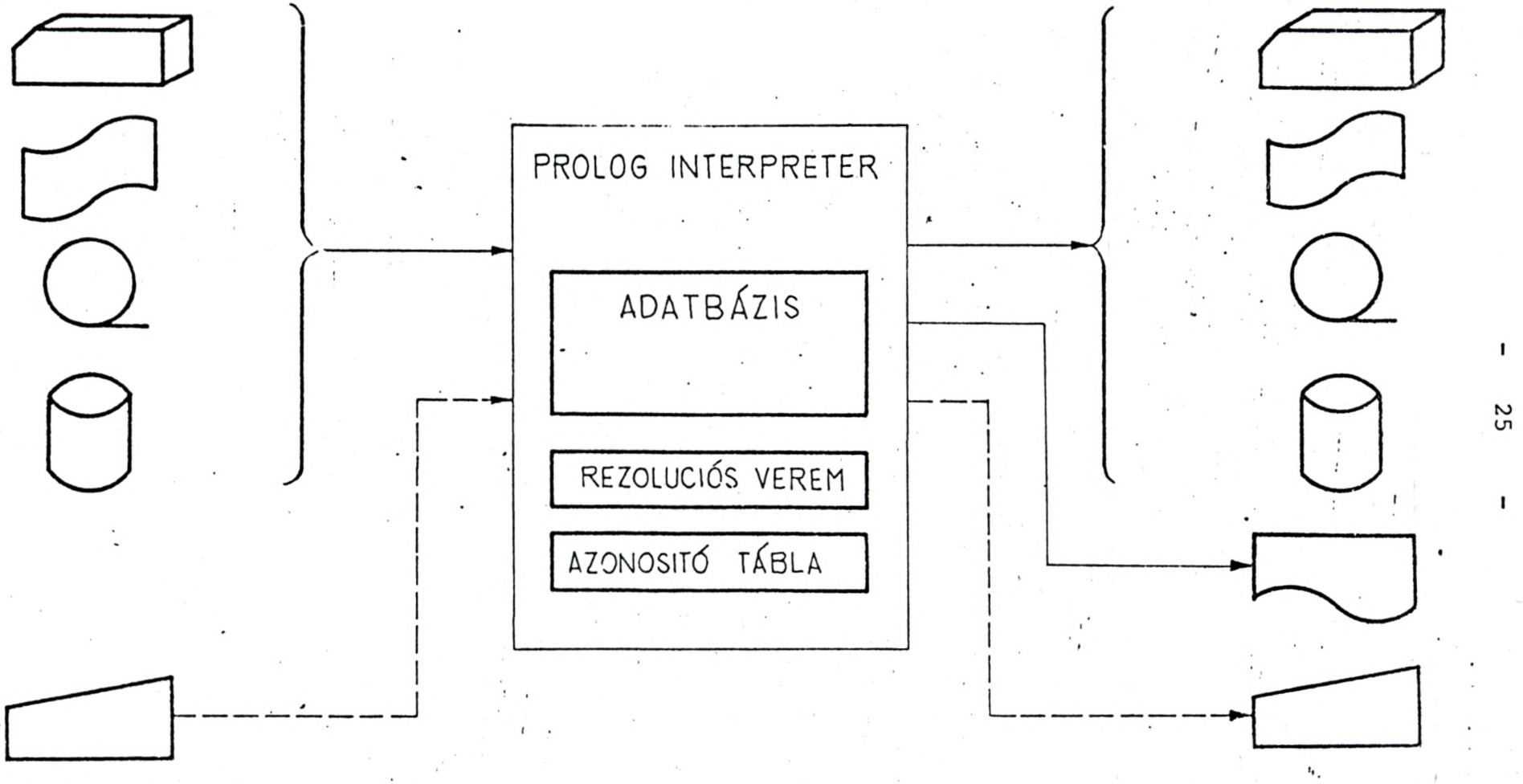
A PROLOG első hazai megvalósitása 1975 májusában készült el, a NIM IGÜSZI-ben CDL nyelven, az ICL 1903A számitógépre. A megvalósitásban felhasználásra került az un. 1. szintü CDL-könyvtár, amely a szimbolikus input/output lehetőségeit és egyéb szimbolumkezelő segédeljárásokat szolgáltatott. Az interpreter ezen változata csak a legfontosabb beépitett /szabványos/ eljárások definicióját tartalmazta. Az 1976-80 években került sor az interpreter továbbfejlesztésére, aminek során kidolgozásra kerültek:

- uj szabványos eljárások,
- nyomkövetési eszközök,
- interaktiv programfuttatást biztositó lehetőségek.

Az interpreter továbbfejlesztésében fontos lépés volt a hangolás elvégzése, amikoris a legsürübben használt kódrészek assembly-szintre való átirásával mintegy 4-szeres gyorsulást sikerült elérni.

A PROLOG interpreter felépitését a következő ábra mutatja:

A PROLOG INTERPRETER FELÉPÍTÉSE



Az interpreter memóriaszervezésében három lényeges táblázat van:

- adatbázis,
- rezolúciós verem,
- azonositótábla.

Az adatbázis a felhasználói programot alkotó logikai állitások /eljárásdefiniciók/ gyüjteménye; ez a futás során tetszőlegesen változtatható eljárások hozzáadásával, ill. elhagyásával.

A rezolúciós verem az egyes feladatok megoldása során szükséges /időleges/ információk tárolására szolgál. A PROLOG interpreter végrehajtási mechanizmusa folytán a rezolúciós verem minden eljárás-hiváskor bővül, de csak a visszalépés során szabadul fel /azaz - szemben a hagyományos rekurziv-eljárás - szervezéssel - egy eljárás sikeres befejeződése után nem szabadul fel a számára lefoglalt hely/.

Az azonositótábla a PROLOG programban szereplő füzérek /string-ek/ és azonositók karakteres alakját tárolja.

2.2 A PROLOG továbbfejlesztése; az MPROLOG

A hazai és külföldi PROLOG implementációkkal szerzett tapasztalatok alapján 1979-80-ban a KSH OSZI támogatásával a NIM IGÜSZI és az SZKI együttmüködésével elkészült a PROLOG rendszer egy uj változata. Az uj, un. MPROLOG rendszer kifejlesztésének célja egy, a réginél hatékonyabb, a felhasználói igényeket jobban kielégitő PROLOG megvalósitás kidolgozása volt. A hatékonyság növelése a PROLOG programok helyigényének és futásidejének csökkentését és a programkészités-, belövés, tesztelés folyamatának meggyorsitását jelenti.

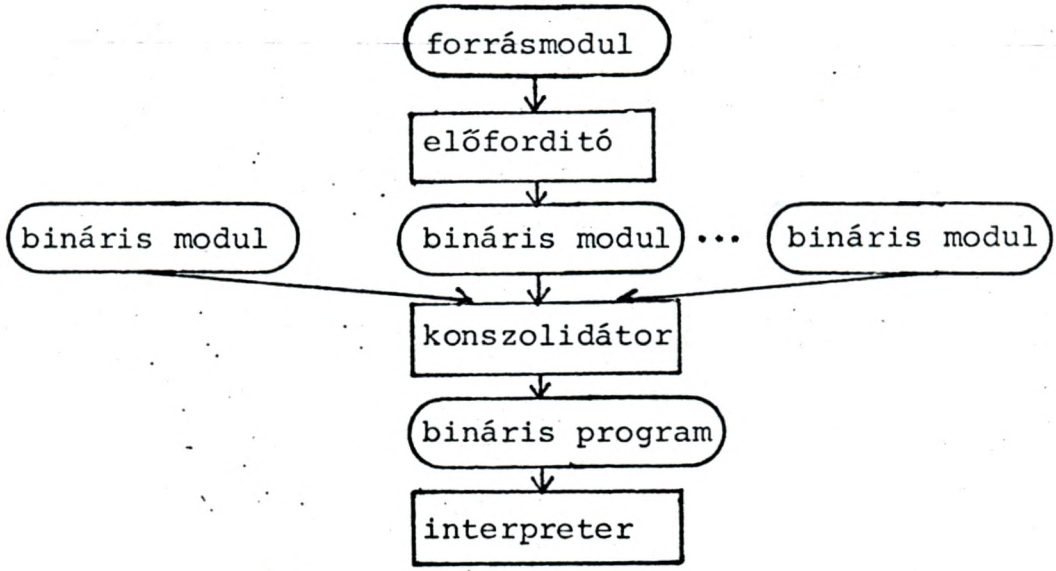
Az MPROLOG moduláris nyelv; a felhasználó programját modulokból szervezheti, ezeket a modulokat egyenként ellenőrizheti. A nyelv szintaxisa a CDL2-éhez közeli, tömör áttekinthető programozást tesz lehetővé. A MPROLOG lehetőséget ad arra, hogy a felhasználó explicit deklarációkkal segitse programja hatékony végrehajtását. A nyelv szabványos lehetőségeket nyujt ahhoz, hogy "célorientált" beépitett eljárásokkal rendelkező interpreter-változatokat hozzunk létre speciális feladatok megoldásához.

Az MPROLOG alaprendszere három CDL2-ben irt programból áll, ezek:

- az előfordító program,
- a konszolidátor és
- az interpreter.

Az előfordító program MPROLOG modulok ellenőrzését végzi, és a forrásalakból tömör, optimalizált belső alakot állit elő.

A konszolidátor egy PROLOG szintü összeszerkesztő program /linkage editor/, amely belső alaku MPROLOG modulokból interpretálható programot állit elő. Az interpreter MPROLOG programok végrehajtását végzi. Az MPROLOG rendszer müködési sémáját a következő ábra mutatja:



Az MPROLOG alaprendszer kidolgozásakor az elsődleges cél a programfuttatás hatékonyságának növelése volt. A programfejlesztés, -belövés segitésére az MPROLOG alaprendszert egy interaktiv programfejlesztő alrendszer egészitiki. Ez már MPROLOG-ban iródott.

A programfejlesztő alrendszer MPROLOG modulok irásához, javitásához, teszteléséhez egyszerü, interaktiv, nyelvorientált /dedikált/ segédeszközöket nyujt. Lehetőséget ad az eljárások nyomkövetésére, még meg nem irt modulok külső kapcsolatainak szimulálására.

A program teszteléséhez az eljárásokban megszakitási pontok definiálhatók, ezeken a pontokon a belövés alatt álló

program futása megszakad, és a felhasználó a programfejlesztő rendszer szolgáltatási segitségével tájékozódhat az eddigi eredményekről, befolyásolhatja a tesztelés további menetét.

2.3 A hazai PROLOG interpreterek installációi

Az 1975-ben az ICL 1903A gépen CDL nyelven elkészült PROLOG interpreter még ugyanebben az évben átvitelre került egy lényegesen eltérő architekturáju gépre, az ICL SYSTEM 4/70-re. Ez az átvitel szinte minden nehézség nélkül lezajlott, és tapasztalatai alapján lehetővé vált az interpreter további installációinak zökkenőmentes lebonyolitása. Az alábbi táblázat felsorolja a szóban forgó PROLOG interpreter hazai installációit, feltüntetve az átvitel évét, az intézményt, számitógéptipust és az operációs rendszert:

1975: NIM IGÜSZI	ICL 1903A /GEORGE-2/
1975: OTSZK	ICL SYSTEM 4/70 /MULTIJOB/
1976: EVIG	EMG 840 /TMIN/
1976: ASZSZ	HWB 66/20 /GCOS/
1976: ELTE	ODRA 1304 /GEORGE 1-2/
1977: MÜM SZÁMTI	ICL 1905 /GEORGE 1-2/
1977: Kőbányai Gyógyszerárugyár	ODRA 1305 /GEORGE 1-2/
1977: EGYT	ODRA 1305 /GEORGE 1-2/
1977: SZKI	SIEMENS 7.755 /BS2000/
1977: SZÁMOK	IBM 370/145 /DOS/VS/
1978: SZÁMKI	R22 /OS/MFT/
1978: SZÁMOK	IBM 370/145 /OS/VS1/
1979: ÉTI-ÉGSZI	SIEMENS 4004 /BS2000/
1979: PMMF /Pécs/	R22 /DOS/
1979: KFKI	R40 /OS/MVT/
1980: SZTAKI	IBM 3031 /CMS/

A felsorolt installációk közül kiemelendő az első interaktiv változat, amely a SIEMENS 7.755 gépen 1977-ben ké-

szült el. Ez a BS2000 operációs rendszer által nyujtott interaktiv környezetre épitve sok uj szolgáltatással könnyitette meg PROLOG programok készitését és belövését. Az operációs rendszer virtuális memóriakezelése a felhasználói tárat jelentősen megnövelte, igy számos korábbi PROLOG-alkalmazás gyakorlatba vétele és hathatós továbbfejlesztése megtörténhetett, további /addig reménytelennek tünő/ alkalmazások kerülhettek kidolgozásra. Ez az interpreter képzete bázisát a hazánkban és a szocialista országokban 1978 után történt IBM-kompatibilis installálásoknak; fejlesztési bázisul szolgál a további PROLOG-fejlesztéseknek is.

Az MPROLOG rendszer az SZKI Siemens 7.755 gépén, a BS2000 operációs rendszer alatt került kifejlesztésre. A rendszer átvitele az IBM/OS operációs rendszer alá már megtörtént, igy az hozzáférhető az EszR család nagyobb gépein is. A rendszer könnyen átvihető minden, CDL2 forditóval rendelkező gépre. Az MPROLOG-installálások táblázata az alábbi:

1980	SZKI	SIEMENS 7.755 /BS2000/
1980	SZTAKI	IBM 3031 /CMS/
1980	NIM IGÜSZI	R22 /OS/MFT/

A nagyszámu PROLOG és MPROLOG installáció a PROLOG és az ujelvü nyelvek iránti egyre növekvő érdeklődést mutatja. Az átvitelek viszonylag könnyü és hatékony végrehajtása elsősorban a CDL /ill. CDL2/ alkalmazásának köszönhető, miután a CDL nyilt nyelv jellege biztositotta a gépfüggetlen és gépfüggő részek könnyen áttekinthető módon való szétválasztását, valamint az utóbbiak esetén a hatékony, rendszerhez alkalmazkodó, gépközeli programozás lehetőségét.

3. Magyarországi PROLOG alkalmazások

A PROLOG első alkalmazásai a tudományos, nem-numerikus programozási feladatok területéről származnak: ezeken a területeken érvényesülhetnek leginkább a PROLOG nagyon magas szintü elemei, lehetővé téve olyan feladatok viszonylag könnyü beprogramozását, amelyek számitógéprevitele hagyományos programozási környezetben nem, vagy csak lényegesen nagyobb ráforditásokkal lenne megvalósitható. A PROLOG külföldi alkalmazásai - miután a PROLOG megvalósitások többsége egyetemi, akadémiai környezetben jött létre - elsősorban az elméleti kutatási témákra szoritkoztak. Igy a legfontosabb külföldi alkalmazások a következők /bővebbet a [Co,80] anyag tartalmaz/:

- természetes nyelv megértése /angol, francia, spanyol, portugál, lengyel nyelvekre/;
- szimbolikus integrálás-deriválás;
- robotok cselekvési tervének generálása;
- geometriai tételbizonyitó programok.

Magyarországon a PROLOG - több körülmény egybeesése folytán - viszonylag hamar több sikeres gyakorlati alkalmazásra talált. Az első sikeres, az érdeklődést felkeltő alkalmazások a gyógyszertervezés és az épitészeti tervezés területén voltak. A kezdeti sikerek arra inspirálták a /PROLOG-alkalmazókkal szorosan együttmüködő/ fejlesztőket, hogy ujabb és ujabb, a felhasználók igényeit egyre jobban kielégitő interpreter-változatokat fejlesszenek ki /interaktiv programfejlesztési környezet biztositása, kényelmes szimbólum-manipulációs szolgáltatások beépitése, stb./. Ezek a fejlesztések természetesen visszahatottak az alkalmazásokra: mélyültek az addigi alkalmazási körök, ujabb területekről jelentkeztek igények.

A PROLOG magyarországi alkalmazásairól a [Sántáné-Tóth,80] összeállitás átfogó képet ad. A következőkben - erre az összeállitásra támaszkodva - témakörök szerint csoportositva felsoroljuk a lényegesebb magyarországi PROLOG-alkalmazásokat. Minden alkalmazásnál megadjuk a realizálás évét, a számitógép/ek/ tipusát, amely/ek/ben az alkalmazás kifejlesztésre vagy adaptálásra került, továbbá az alkalmazáshoz kötődő irodalmat.

3.1 Gyógyszerkutatás

3.1.1 PEPTIDEK BIOLÓGIAI HATÁSÁNAK ELŐREJELZÉSÉRE SZOLGÁLÓ JELLEMZŐK KISZÁMITÁSA

1979; SIEMENS 7.755, [Darvas,80b].

A peptidek a fehérjékhez hasonló, nagy fontosságu vegyületek, amelyek egyre jelentősebb helyet foglalnak el a gyógyszerkutatásban. A vegyületek biológiai hatásának előrejelzése egy olyan - mindeddig kidolgozatlan - modellt igényel, amely a peptidek egyéb szerves vegyületektől eltérő, speciális tulajdonságait emeli ki.

A szóbanforgó PROLOG-FORTRAN vegyesnyelvü programrendszer ilyen modell kidolgozását segiti elő, és egyben peptidek biológiai hatásának előrejelzésére szolgál.

A rendszer a peptidek kémiai tulajdonságaiból kémiai szerkezeti elemeket generál, majd a kémiai szerkezeti elemekhez számszerü jellemzőket rendel hozzá. Ezeket a feladatokat PROLOG nyelven irt programok látják el. A számszerü jellemzők és a peptidek biológiai hatása között FORTRAN nyelvü programok keresnek összefüggést.

3.1.2 GYÓGYSZERIPARI KUTATÁSSZERVEZÉS

1979; SIEMENS 7.755, —

A programrendszer gyógyszeripari kutatás-szervezési problémák megoldásában nyujt segitséget, adatok visszakeresése és automatikus következtetés segitségével. A programrendszer gyógyszer- és növényvédőszer hatóanyagok uj felhasználási lehetőségeit következteti

ki automatikusan és interaktiv módon. A rendszer PROLOG és EDT /a BS2000 szöveg-szerkesztője/ nyelven készült programokat tartalmaz; gyógyszeripari megrendelésre készült.

3.1.3 VEGYÜLETEK GYÓGYHATÁSÁNAK ELŐREJELZÉSÉRE SZOLGÁLÓ FIZIKOKÉMIAI TULAJDONSÁGOK KISZÁMITÁSA

1976-78; ICL 1903A, ODRA; [Darvas,78a], [Darvas,78d].

A számitógépes hatóanyag-tervezés /gyógyszerek, növényvédőszerek hatóanyagának tervezése/ során a számitási módszerek jelentős része a vegyületek "zsiroldhatóságát", lipofil jellegét megadó fizikokémiai tulajdonságból, az un. logP értékből indul ki. Ennek az értéknek a kiszámitása kézzel bizonytalan pontosságu, és hosszadalmas feladat. A PROLOG program irása idején /1976-ban/ mindössze egy számitógépes program volt ismeretes a szakirodalomból a számitás elvégzésére.

3.1.4 GYÓGYSZERINTERAKCIÓ-ELŐREJELZÉSE

1975; ICL 1903A, ODRA; [Darvas,76a], [Darvas,78c], [Darvas,79b], [Futó,78a].

A gyógyszerek együttes adásakor keletkező hatásmódosulások - mellékhatások fellépése, a hatóanyag vérszintjének csökkenéséből vagy növekedéséből adódó jelenségek /más néven gyógyszerinterakciók/ - a klinikai és farmakológiai szintü gyógyszeres kezelés egy fontos problémakörét alkotják.
A kidolgozott programrendszer a gyógyszer-hatóanyagokat alkotó vegyületek fizikokémiai, farmakológiai és kémiai tulajdonságaiból kiindulva próbál a gyógyszerinterakciók lehetőségére következtetni.

3.1.5 LIGANDKÖTŐ RENDSZEREK NUMERIKUS ANALIZISE
1979; SIEMENS 7.755; [Kőfalusi, 79c].

A kismolekuláju anyagokat /ligandokat/ kötő makromolekuláris rendszerek analizise fontos biokémiai feladat: nem nélkülözheti pl. a gyógyszerkutatás, az immunológia, az enzimológia, az endokrinológia, stb. A ligandkötés jellemzésére a kötött/szabad ligandkoncentráció-arányt használják. Megfelelő matematikai egyenletek irják ezt le: ezen egyenletek állandóit nemlineáris regresszóval, a legkisebb négyzetek elve alapján szokták meghatározni. A regresszó hatékonysága sokban mulik a "jó" kezdőértékek megválasztásán. Minthogy ezt eddig még nem sikerült algoritmizálni, a programozó csupán "jó" intuiciójában bizhat. Egy szimbolikus differenciálást lehetővé tevő /FORTRAN SUBROUTINE szegmenst generáló/ PROLOG program felhasználásával biztositható a ligandkötő rendszerek numerikus analiziséhez szükséges jó kezdőértékek automatikus beállitása. A program a 3.7.2 pontban ismertetett programgenerátor felhasználásával készült.

3.1.6 ÖNREPRODUKÁLÓ BIOKÉMIAI FOLYAMATOK TESZTELÉSE ÉS MODELLEZÉSE
1979; SYSTEM 4/70, SIEMENS 7.755; -

A programmal tetszőleges biokémiai önreprodukáló ciklusok analizálhatók és modellezhetők. A programba beépitett adatbázis cseréjével tetszőleges ciklus vizsgálható. A bemenő adatokat a formális reakcióegyenletek adják; a feldolgozás során kiválasztásra kerülnek a tápanyagok, a végtermékek és

az attraktorok. A program jó példa a strukturális rendszermodelleknek PROLOG-ban való gyors és kényelmes definiálására; ezen előnyök a PROLOG által kezelhető logikai kifejezések használatából adódnak.

3.1.7 ENZIMSZEKVENCIÁK HASONLÓ SZUBSTRUKTURÁINAK MEGKERESÉSE

1979; SIEMENS 7.755; [Mátrai, 79].

Az ismert enzimcsoportokon belüli homológiák, strukturális, funkcionális és evolúciós hasonlóságok, illetve különbségek felfedése eddig - a technikai lehetőségek hiánya miatt - nem volt megvalósitható: a szokásos algoritmusok ugyanis olyan lassuak, hogy gyakorlatilag megvalósithatatlanok. Az eddig publikált megoldásoknál általánosabb és rugalmasabb megoldáshoz lehet jutni, ha az enzimeket lineáris gráfoknak tekintjük; ilyen gráfok hatékony kezelését PROLOG programmal már lehet biztositani.

A kidolgozott program ismert szekvenciáju és hasonló müködésü enzimek funkció szempontjából várhatóan releváns szubstrukturáinak megkeresésére szolgál. Tetszőleges nagyságu és tetszőleges számu hibahellyel rendelkező, hasonló primer szekvencia-egységek megtalálására alkalmas.

3.2 Információ-visszakereső rendszerek

3.2.1 EGY LOGIKAI ALAPU KÉMIAI INFORMÁCIÓS RENDSZER

1976-78; ICL 1903A, ODRA, SIEMENS 7.755
[Darvas, 78b, 79a].

Kémiai strukturák számitógépes kezelésének alapproblémája, hogy a matematikailag irányitatlan gráfoknak megfelelő strukturák csak nehézkesen tárolhatók és kezelhetők a szokásos adatfeldolgozási módszerekkel. Tetszés szerinti /szubstruktura-/ visszakeresést biztositó rendszerek kifejlesztése aktuális téma; a szakirodalomban azonban eddig még csak néhányat közöltek.

A programrendszer kémiai strukturák és struktura-részletek /szubstrukturák/ visszakeresését teszi lehetővé. A programokat ki lehet egésziteni a strukturákhoz és a struktura-részletekhez, mint feltételekhez csatlakozó olyan logikai állitásokkal, amelyek a strukturákat tartalmazó vegyületek kémiai, ill. biológiai tulajdonságaira vonatkoznak. A programrendszer teljes kiépitése 1983-ra várható.

3.2.2 LEVEGŐTISZTASÁGVÉDELEMMEL KAPCSOLATOS INTERAKTIV INFORMÁCIÓ ELLÁTÓ RENDSZER

1977; ICL SYSTEM 4/70; [Bendl,79a], [Futó, 78a].

A programrendszer Magyarország megyéire és Budapestre nézve hét ipari légszennyező anyag alapkoncentrációját kezeli, megyénként 15-20 körzetre osztva. A területi besorolásnak megfelelően ellenőrzi, hogy a müködő vagy tervezett létesit-

mény légszennyezése a megengedett alatt van-e. Amennyiben nem, ugy kiszámitja a koncentráció csökkentéséhez szükséges kéménymagasságot, és adatbázisa alapján ajánlatot tesz az iparágnak és a technológiának megfelelő ipari szürőberendezésére is. A programrendszer interaktivan működik; tervezésénél alapvető szempont volt, hogy mind vezetői, mind tervezői, mind pedig kutatói beállitottságu emberek fel tudják használni.

3.2.3 NÖVÉNYI KÁRTEVŐK ÉS NÖVÉNYVÉDŐSZEREK ADATAIT KEZELŐ INFORMÁCIÓ VISSZAKERESŐ RENDSZER

1977; ICL SYSTEM 4/70; [Futó, 78a].

A különböző növényvédőszerek adott környezetben várható hatásának vizsgálata céljából készitett PROLOG programmal a következő három tényező kölcsönhatását lehet vizsgálni:

- egy kultura ártalmai: betegségek, rovarkártevők, stb.;
- adott betegségek, rovarkártevők, stb. elleni védőszerek; növényvédő-, rovarirtó szerek, stb.;
- azok a kulturák, amelyekben adott növényvédő-, rovarirtó szerek, stb. felhasználhatók.

A rendszer - interaktiv módon - a következő fajta kérdésekre tud válaszolni:

- a háromféle tényezőnek megfelelő adatok kilistázása;
- két kiválasztott tényezőhöz a hozzá kapcsolható harmadik tényező meghatározása;
- adott kultura-hatóanyag-kártevő hármashoz a kulturában a kártevő ellen bevethető rovarirtószerek meghatározása.

3.2.4 AZ "ANSWER" SOFTWARE FEJLESZTŐ RENDSZER INFORMÁCIÓS ALRENDSZERE

1977-80; ODRA 1304, SIEMENS 7.755; [Bán,79].

Az ANSWER rendszer a software-fejlesztő munka gépesitésére szolgáló programfejlesztő eszköz; jelenlegi változata CDL2 nyelvü programok kifejlesztését teszi lehetővé. A rendszer több alrendszerből áll; ezek közül igen fontos a szóbanforgó információs alrendszer, amelynek MPROLOG-ban történő implementálása jelenleg folyamatban van. Az információs alrendszer feladata többek között a CDL2 program moduljai közötti kapcsolódások ellenőrzése, egyes CDL2-objektumok módositása hatáskörének vizsgálata, adott feladatot ellátó CDL2-modul megkeresése.

3.3 Építészeti tervezés

3.3.1 EGYSZINTES CSARNOK MÉRETEZÉSE ELŐREGYÁRTOTT PANELEKBŐL

1975; ICL 1903A; –

A program az első magyarországi PROLOG-alkalmazás; előregyártott elemekből készülő, egyszintes daruzatlan ipari csarnokot tervez. Az alaprajz téglalap alaku, kétszeresen szimmetrikus. A szerkezet oszlopokból, azokban fekvő kéttámaszu gerendákból és a gerendákban fekvő, téglalap alaku födémpanelekből áll. A rendelkezésre álló előregyártott elemek adatait /geometriai méretek, önsuly, teherbirás/ a programba tényállitások formájában kell beépiteni.

A kiinduló adatok a csarnok geometriai méretei, és a födémet terhelő egyenletesen megoszló teher intenzitása. A program megtervezi az alaprajzi rasztert /födémpanel kiosztást/, és kiválogatja a geometriai és statisztikai feltételeknek megfelelő elemeket.

3.3.2 PANELES LAKÓÉPÜLETEK ÉPITÉSZETI TERVEZÉSE

1976; ICL 1903A, R40; [Márkusz, 77a, 77b].

A kidolgozott PROLOG program /az adatbázisba beépitett/ panelelemekből adott méretü, szobaszámu és félszobaszámu lakások alaprajzi variánsait tudja előállitani. A program a megadott adatokból előállitja az összes lehetséges lakás alaprajz variánst. A 3.3.3-ban leirt programrendszer jelen program egy kiterjesztése.

3.3.3 TÖBBSZINTES LAKÓÉPÜLETEK TERVEZÉSE

1980; SIEMENS 7.755, IBM 3031; [Márkusz, 80a, 80b].

A programrendszer felhasználásával többszines lakóépületek tervezése a következőképpen történik. A rendszer először is előállitja az egyes felhasználók /mint a leendő lakástulajdonosok/ speciális igényeit kielégitő összes lehetséges, a megadott alapelemekből összeállitható alaprajzi variánsokat. Ezután a felhasználók ezek közül kiválasztják a számukra legkedvezőbbeket, és kizárják a nemkivánatosokat. A programrendszer ezután az egyes felhasználók által kedvezőnek minősitett összes variánsból olyan lakóépület-tervet generál, amely eleget tesz az épülettel szemben támasztott funkcionális és geometriai követelményeknek. Az épületben minden leendő lakástulajdonos számára van egy olyan lakás, amely megfelel a saját igényeinek.

A rendszernek az az előnye, hogy sok /pl. 15-20/ különböző alaprajzu lakásból válogat össze olyan épületeket, amelyek megfelelnek a gyakorlati kivitelezés követelményeinek.

3.3.4 AUTOMATIZÁLT SZOLITER-ALAPTERVEZÉS

1979; ICL SYSTEM 4/70; SIEMENS 4004, [Holnapy,79].

A program által megoldott feladat a következő: épületpillérek alá megfelelő alaptestek beválogatása tényállitásokkal definiált készletből, azaz eleve adott rendszerkomponens-készletből. A lehetséges megoldásokból a program különböző szempontok

szerint optimalizált változatokat is ki tud választani. /Ilyen célkitüzés lehet pl. "egyszerübben szerelhetőnek lenni"./

Kisérleti próbafutásnak van alávetve a program egy olyan továbbfejlesztett változata, amelynél tetszés szerinti erőrendszer /erőlista/ és távolságlista megadása lehetséges a célállitásban, és az erők helyén alkalmazandó alaptestek azonositói képezik a végeredményt.

Jelenleg tervezés alatt van olyan PROLOG program kidolgozása, amely épitészeti müszaki tervezés válogatás alapján történő kisérleti megvalósitására fog lehetőséget adni.

3.4 Software

3.4.1 ADAT-ÉRVÉNYESSÉG VIZSGÁLATÁRA SZOLGÁLÓ COBOL PROGRAMOK GENERÁLÁSA

1978; SIEMENS 7.755, ICL 1903A; [Láng,78].

A PROLOG nyelven irott programgenerátor olyan ANSI-COBOL programok előállitására alkalmas, amelyek feladata az input adatok ellenőrzése. A generált program a hibátlanokat kiirja egy output filera, a hibásakat /a hiba okának megjelölésével/ kilistázza. A generált COBOL program által kezelt file-ok felépitése és az ellenőrzési szempontok terminálról vagy háttér file-ból beadott paraméterek alapján határozhatók meg.

3.4.2 EGYSZERŰ ADATFELDOLGOZÓ COBOL PROGRAMOK GENERÁLÁSA

1979; SIEMENS 7.755; [Futó,79d].

A programgenerátorok az SZKI-ban használatos szabványoknak eleget tevő COBOL programokat generálnak. Ez a szabvány a COLAMI szabvány, amely a COLUMBUS strukturált precompiler és az AMIGO programozási konvenciók ötvözete.

A generátorok az alábbi tpusu adatfeldolgozási feladatokhoz generálnak programokat: adatállomány listázása, adatállomány karbantartása, két adatállomány összeválogatása, valamint elsődleges input adatállomány ellenőrzése.

A generált COBOL programok által kezelt input és output adatállományok strukturája, valamint az

egyes programok által elvégzendő feladatok terminálról vagy file-ról beadott egységes paraméterek alapján határozhatók meg.

3.4.3 INTEL 3000 MIKROPROGRAMOK BEÜLTETÉSE

1978; SIEMENS 7.755; [Szeredi J,78], [Garami,78].

Az INTEL 3000 mikroprocesszor speciális cimzési mechanizmussal rendelkezik. A program-tárat egy olyan mátrixnak lehet elképzelni, amelynek minden eleme egy olyan utasitás, amely tartalmazza a rákövetkező utasitás cimét /azaz helyét a mátrixban/. Egyes mikroutasitás-fajtákra azonban pontosan rögzitettek azok az alternativák, amelyek az utána végrehajtandó utasitás helyét /abszolut és/vagy utasitásrelativ módon/ megadják. Ilyen korlátozás pl.: a rákövetkező utasitás a szóbanforgó utasitással azonos oszlopban legyen. A programozóra hárul a cimkiosztás, amely igen fáradságos, többszöri kisérletet igénylő feladat. Ennek hátránya nemcsak a mikroprogramok tervezésénél, hanem karbantartásánál is jelentkezik. A PROLOG program - a mintaillesztések révén - gyorsan figyelembe tudja venni az egyes mikroutasitásokra vonatkozó megkötéseket, sikertelen beültetési kisérlet végrehajtása után pedig az uj kisérletet - a visszalépési algoritmus segitségével - igen hatékonyan meg tudja oldani.

A program inputja egy részben már betöltött tár, és a beültetésre szánt program. Ezek alapján a program - racionális gépidő ráforditással - vagy

egy lehetséges beültetési képet ad outputra, vagy kijelzi az adott program beültetésének lehetetlenségét.

3.4.4 PROGRAMOZÁSI STILUS ÉS HATÉKONYSÁG ELEMZÉSE

1979-80; IBM 370/145; [Gerő,80].

A programrendszer szintaktikusan helyes PL/I és COBOL nyelvü programok minőségének következő szempontok szerinti analizisét végzi el: programstruktura, programozási stilus, program-hatékonyság és komplexitás. A programrendszer az analizis során általa felfedett hibák elkerülése céljából megoldási alternativákat javasol a programszerkezet megváltoztatására. E strukturális analizis során a rendszer meghatározza és hierarchia diagramok formájában kinyomtatja a program logikai strukturáját, megjelölve az általa végrehajtott strukturális korrekciókat. A programrendszer hierarchia diagrammal foglalkozó tagja optimalizáló PL/I-ben, többi tagja PROLOG-ban lett implementálva. A rendszert a SZÁMOK-ban folyó oktatás számitógéppel történő támogatására használják.

3.4.5 PROLOG PROGRAMELLENŐRZŐ RENDSZER

1977-78; ICL SYSTEM 4/70, SIEMENS 7.755. [Balogh,77].

Az ellenőrző rendszer PROLOG nyelven irt programok szemantikus ellenőrzését végzi; a parciális helyesség bizonyitására használható. A formulatranszformáló és az általános tételbizonyitó program által alkotott részrendszer azonban önmagában is alkalmazható interaktiv tételbizonyitásra. Az interaktiv

formula-transzformáló program természetes dedukciót végez, beépitett vagy a párbeszéd során kapott transzformációs /következtetési/ sémák alapján. A program az adatként kapott sémákat interpretálja. A sémák tömörek és szemléletesek, az alkalmazandó transzformációt és az alkalmazás módját a felhasználó intuiciója alapján választhatja meg. Az általános tételbizonyitó program rezoluciós elven müködik.

A rendszer kisérleti jellegü. A [Balogh, 78a] és a [Balogh, 79a] dolgozatok a rendszerrel kapcsolatos elvi kérdésekkel is foglalkoznak.

3.4.6 DOKUMENTUMOK KÉSZITÉSÉT SEGITŐ RENDSZER

1980; SIEMENS 7.755; [Fidrich, 80a, 80b].

A dokumentumok készitését segitő rendszer rendeltetése a szöveges objektumok - elsősorban a szabványos programdokumentumok - készitésének segitése. A rendszer szemléletében a szöveges objektumokat különböző tipusu követelményekből felépülő követelményrendszerek határozzák meg. A követelmények vonatkozhatnak az objektumok felbontására, tartalmára, sorszámára, stb. A rendszer az objektumot meghatározó követelményrendszer szigoritására interaktiv módon ad lehetőséget.

3.4.7 SOFTWARE ÉS HARDWARE OBJEKTUMOK TERVEZÉSE

1978; ICL 1903A, SIEMENS 7.755; [Balogh,78b,78c] [Farkas,78], [Garami,78], [Herényi,78].

A PROLOG-alkalmazások tapasztalatai megmutatták, hogy a nyelv igen alkalmas - más nyelveken nem,

vagy csak nehezen megoldható - feladatok megoldására. A PROLOG-ban irt programok igen világosan tükrözik a feladat szerkezetét és a megoldás utját.

Ez - valamint a PROLOG-kód irása közbeni ellenőrző futtatások lehetősége - adta az ötletet arra, hogy kisérletek történjenek a PROLOG-nak tervezési nyelvként való felhasználására. Ennek során sor került
- rendező-programcsalád [Balogh,78b],
- ANSWER file-kezelő rendszer [Farkas, 78],
- modul-könyvtár kezelő [Herényi,78],
- RPG-elemző [Herényi,78],
- INTEL 3000 mikroprogram beültető program [Garami,78]

tervének elkészitésére.

A fenti kisérleti alkalmazások azt mutatták, hogy a PROLOG alkalmas software /és hardware/ objektumok tervezésére; hiányoznak azonban a nyelvből az adatkezelési lehetőségek, és az "igazi" interaktiv tesztelési segédletek, valamint a tervezésre való alkalmazáshoz ajánlható módszer. A tanulságok figyelembevételével került kifejlesztésre egy logikai-alapu tervező nyelv/módszer/rendszer, az LDM, amelyről rövid ismertetést a [Balogh,79b] és [Szeredi P,80] dolgozatok tartalmaznak.

3.5 Architektura tervezés

3.5.1 ETALON-PROGRAM GENERÁTOR NYELVI ARCHITEKTURÁK VIZSGÁLATÁRA

1978; SIEMENS 7.755; [Kiss V. 78].

Valamely magasszintü programozási nyelvet támogató számitógép-architektura tervezésének és vizsgálatának különböző fázisaiban jó hasznát lehet venni olyan, az adott nyelven irt kisméretü, lefuttatható mintaprogramoknak, amelyek előre rögzitett, a nyelv felhasználási gyakorlatát tükröző statisztikai tulajdonságokkal rendelkeznek. /Ilyen tulajdonságok pl. az egyes utasitásfajták és/vagy adattipusok aránya./ Az ilyen etalonprogram-változatok sokaságának elkészitése megkivánja azok gépi uton történő előállitását. Kisérletképpen elkészült egy olyan, SPL /Simple Programming Language/ nyelvü etalon-programokat generáló PROLOG program, amelynek inputja az SPL szintaxisa, valamint a generálandó programoktól elvárt statisztikai jellemzők.

3.5.2 KIÉRTÉKELŐ SZIMULÁTOR MAGASABBSZINTÜ ARCHITEKTURA TERVEZÉSÉHEZ ÉS KISÉRLETI ELLENŐRZÉSÉHEZ

1979; SIEMENS 7.755; [Kiss V. 79].

A kiértékelő szimulátor kifejlesztésének alapvető célja a nyelvorientált számitógépek architektura tervezési folyamatának gépi segédeszközökkel való támogatása. A tervezési folyamat több fázisát kivánja támogatni; alkalmas

- a vizsgált architektura hatékonyságára jellemző mennyiségi mutatók mérésére etalon benchmark programok futtatása révén /mérhető jellemzők: programméret, utasitás lehivás szám, operativ tár hivatkozások száma, operativ tár - processzor adatátvitelek, továbbá regiszter tár - processzor adatátvitelek mennyisége/,

- a specifikált architektura helyességének kisérleti ellenőrzésére, teszt programok futtatása utján, végül

- a forrásnyelv használatának környezetére jellemző dinamikus statisztikák mérésére /egyéb hatékonyabb eszközök hiányában/.

A közismert "Flynn-féle" Kanonikus Értelemzési Forma hatékony alkalmazhatósága, valamint a hazai fejlesztési igények szempontjait figyelembevéve, a szimulátor a COBOL forrásnyelv Kanonikus Értelmezési Formája szerinti architekturákhoz készült.

3.6 Szimuláció

3.6.1 A T-PROLOG UJELVÜ SZIMULÁCIÓS NYELV INTERPRETERE

1980; SIEMENS 7.755; [Futó, 80b, 80c].

A T-PROLOG egy logikai alapu, ujelvü szimulációs nyelv, amely lehetőséget nyujt párhuzamosan, egyidejüleg végrehajtásra kerülő feladatok /"bizonyitások"/ menetének koordinálására. Ennek megfelelően a nyelv interpretere /amely a PROLOG nyelv interpreterének felhasználásával került realizálásra/ egy PROLOG-szerü tételbizonyitón kivül rendelkezik egy belső óra mechanizmussal is. Ez a belső óra a hagyományos szimulációs nyelveknél /pl. SIMULA-nál, GPSS-nél/ látottakhoz hasonlóan müködik. A "párhuzamositást" egy egyszerü interaktiv algoritmus végzi, amely beépitett eljárásokra támaszkodó stratégián alapul; e beépitett eljárások lecserélésével /a felhasználó által is/ módositható az ütemezési stratégia.

A T-PROLOG ujelvü szimulációs nyelv, ahol az ujelvü jelző a "MIT és nem HOGYAN" elvre utal. A T-PROLOG-ban ugyanis nem kell a szimulációs modellt pontosan leirni, hanem a szimulálandó folyamat bizonyos eredményeinek előirása alapján a rendszer maga keresi meg a modell jellemző paramétereit /lényegében véve a PROLOG mintaillesztési és visszalépési mechanizmusa segitségével/.

A T-PROLOG a VÁTI hosszu- és nagytávu regionális modelljeinek vizsgálatánál kerül először alkalmazásra.

3.6.2 TÁVADATFELDOLGOZÁSI HÁLÓZATOK MODELLJEINEK GENERÁLÁSA

1980; SIEMENS 7.755; -

A programgenerátor távadatfeldolgozási hálózatok modelljeinek generálását támogatja: GPSS /SIAS/ nyelvü modell-leirás generálására szolgál. A felhasználó a generátorral való párbeszéd utján adja meg a hálózat leirását /topológia, vonali vezérlő algoritmusok, a csatornák tipusa, stb./. A generált modell ellenőrzi a rendszer átbocsátó képességét, az egyes csomópontokon várakozó sorokat, stb.

3.7 Egyéb alkalmazások

3.7.1 MAGYAR NYELVÜ SZÖVEGEK SZÁMITÓGÉPES MORFOLÓGIAI VIZSGÁLATA

1979; SIEMENS 7.755; [Kiss Z.79].

Magyar nyelvü szövegek számitógéppel történő elemzése - a nyelvtan bonyolultsága, különösen pedig a nyelv agglutinációs jellege miatt - máig nem megoldott feladat. A különböző igealakok, szórendi kapcsolatok, magán- és mássalhangzó-hasonulások, stb. miatt igen nehéz algoritmizálni a feladatot. Az elemzés nagyszámu próbálgatást igényel, ami kinálja a PROLOG alkalmazását. Az eddig elkészült program a szövegelemzés első lépését, a szövegek morfológiai vizsgálatát végzi. A program tulajdonképpen két automata: egyik a magyar igealakok, a másik a toldalékolt főnevek morfológiai elemzését végzi.

3.7.2 TÖBBVÁLTOZÓS VALÓS FÜGGVÉNY ELSŐ N FORMÁLIS DERIVÁLTJÁNAK ELŐÁLLITÁSA

1979; SIEMENS 7.755; [Kőfalusi, 79b].

A PROLOG program szimbolikus differenciálás utján előállitja a megadott többváltozós összetett függvény első n deriváltját. /A függvény lehet igen nagy komplexitásu./ A szimbolikus deriváltak felhasználásával a program ezután generál egy, a deriváltak helyettesitési értékeit kiszámitó FORTRAN szubrutint, végrehajtva eközben a lehetséges egyszerüsitéseket és kiemeléseket.

A szubrutin generálását végző PROLOG program a szerző által kidolgozott, és először e programban fel-

használt gráffal reprezentálható állapottér fogalmára, mint egy lehetséges uj PROLOG-elemre épit /lásd [Kőfalusi,80]/.

3.7.3 EGYSZERÜSITÉS MATEMATIKAI STRUKTURÁKBAN 1979-80; SIEMENS 7.755/; [Kőfalusi,79b].

A fejlesztés alatt álló program matematikai strukturák igen széles osztályában /csoportok, gyürük, testek, Boole-hálók, stb./ megengedett kifejezéseket képes egyszerüsiteni. A bináris faként ábrázolt kifejezésekben alulról-felfelé, jobbról-balra haladva egyszintü előre- és többszintü hátranézéssel végzi a feldolgozást. Asszociativ operátorlánc esetén a megfelelő sorrendben rendez, a müveleti szabályok alapján elvégzi az összevonásokat és egyszerüsitéseket, majd n-argumentu prefix strukturává alakit, amelyen további egyszerüsitéseket végez. A program képletrendezésre, bizonyos relációk kiértékelésére is alkalmas.

4. PROLOG-alapu rendszerek - fejlesztési célkitüzések

A PROLOG alkalmazási sikerei két olyan fejlesztés kiindulópontjául szolgáltak, amelyek a PROLOG rendszerre alapozva valamely speciális területen nyujtanak "ujelvü", "nagyon magas szintü" szolgáltatásokat. Ez a két speciális terület a szimuláció, és a programtervezés; ezek a T-PROLOG /amelynek interpretere a PROLOG interpreter felhasználásával készült, lásd 3.6.1/ és az LDM /amelynek első kisérleti realizálása MPROLOG-ban készült/.

A következőkben rövid ismertetését adjuk a TPROLOG és az LDM nyelveknek/rendszereknek, majd vázoljuk a PROLOG-gal kapcsolatos további fejlesztési célkitüzéseket.

4.1 Ujelvü szimuláció - T-PROLOG

1979-80-ban kidolgozásra került a PROLOG-nak párhuzamos feldolgozási lehetőségeket biztositó, továbbfejlesztett változata, a T-PROLOG. A nyelv lehetőséget nyujt a párhuzamosan, egyidejüleg végrehajtásra kerülő feladatok /"bizonyitások"/ menetének koordinálására. Ennek megfelelően egy PROLOG-szerü tételbizonyitón kivül rendelkezik egy belső óra mechanizmussal is, amely a hagyományos szimulációs nyelvekhez /SIMULA, GPSS/ hasonlóan müködik. A "párhuzamositást" egy egyszerü interaktiv algoritmus végzi, amely beépitett eljárásokra támaszkodó stratégián alapul; e beépitett eljárások lecserélésével /a felhasználó által is/ módositható az ütemezési stratégia.

A T-PROLOG /a fent felsorolt tulajdonságok, valamint a PROLOG-tól örökölt tulajdonságok alapján/ tulajdonképpen egy "ujelvü" szimulációs nyelv /itt az "ujelvü" jelző a "MIT és nem HOGYAN" elvre utal/. T-PROLOG-ban ugyanis nem kell magát a szimulációs modellt pontosan leirni, hanem a szimulálandó folyamat bizonyos eredményeit előirva, a rendszer maga keresi meg a modell pontos paramétereit /lényegében a PROLOG mintaillesztési és visszalépési mechanizmusa segitségével/.

A T-PROLOG a VÁTI hosszu- és nagytávu regionális modelljeinek vizsgálatánál kerül először alkalmazásra.

4.2 Programtervezés logikai alapokon - LDM

1978-80 során a PROLOG-alkalmazások tapasztalaiból kiindulva került kifejlesztésre az LDM /Logic-based Development Method/. Az LDM nyelv a korszerü tervező-, és a hagyományos magasszintü nyelvi követelményeknek egyaránt eleget tevő adatkezelési lehetőségekkel, valamint algoritmikus szerkezetekkel bővitve, egységes keretben ad eszközöket software tervezéshez. E tervező nyelv biztositja a felhasználó számára, hogy egy feladat /leiró jellegü/ specifikációjából kiindulva, ujabb és ujabb tervrészletek beiktatásával, azok algoritmikusabb tervrészletekre való lecserélésével, továbbá a feladat specifikációjának a tervbe ujként bevezetett tervrészleteknek megfelelő bővitésével jusson el a feladat megoldásának egy algoritmikus jellegü leirásához. A megvalósitás kisérleti stádiumában van egy olyan rendszer létrehozása, amely - megfelelő tervezési módszert sugallva - kétféle módon kivánja a tervezőt támogatni az előbbiekben vázolt tervezési lépések végrehajtásában. Először ugy, hogy egy-egy tervezési lépés során állandóan ellenőrzi a felhasználó munkáját, másodszor pedig ugy, hogy minden egyes lépés megtétele után biztositja a feladat /ill. a feladat megoldása/ ujként előállt leirásának, tervszintjének futtatását. Az LDM-rendszer 1983-ra fog elkészülni. A kisérleti realizálással egyidőben a nyelv próba-alkalmazásai is beinditha tók. A rendszer elsőként CDL-szerü programterv készitésére lesz alkalmas, igy a jelenlegi ANSWER rendszer tervező alrendszereként fog funkcionálni. A későbbi fejlesztések során egyéb nyelvü tervek készitésére is alkalmassá tehető az LDM.

4.3 Fejlesztési célkitüzések

A PROLOG fejlesztési célkitüzések - az ismertetett PROLOG-alapu rendszereken tulmenően - az MPROLOG megvalósitáshoz kapcsolódnak. A további fejlesztésre alapvetően az alábbi három területen van szükség:

/1/ interaktiv programozási környezet létrehozása,
/2/ konkrét alkalmazásokhoz kapcsolódó beépitett eljárások, más nyelvekkel való kapcsolat megvalósitása,
/3/ MPROLOG forditóprogram kifejlesztése.

Az interaktiv programozási környezet az ANSWER programfejlesztő rendszerhez hasonló szolgáltatásokat kell nyujtson. Ezen szolgáltatások egy korlátozott részhalmazát a jelenlegi MRPOLOG interaktiv programfejlesztő alrendszer biztositja, de szükséges ennek további szolgáltatásokkal való bővitése /interaktiv próbapad, dedikált szerkesztő, stb./.

A másik fejlesztési igény egyes olyan alkalmazások oldaláról jelentkezik, amelyek PROLOG-ban csak rossz hatékonysággal beprogramozható funkciókat használnak. Ezek az alkalmazások más, algoritmikus programnyelven irt részek beillesztését igénylik. Az MPROLOG rendszerben kialakitásra került a PROLOG és CDL2-eljárások közötti szabványos interface /ezt használják a beépitett eljárások is/. Ennek az interface-nek a felhasználásával megkezdődhet a legfontosabb alkalmazások hangolása, azaz a legsürübben használt PROLOG-programrészek CDL-szintre való átirása. Ugyanezen interface-en keresztül, a CDL2 nyilt nyelv jellegét kihasználva kerülhet sor más programnyelveken irt

szubrutinok PROLOG-ból való hivási lehetőségének biztositására.

Az MPROLOG forditóprogram létrehozására irányuló fejlesztő-munka már 1980-ban beindult: elkészült a forditóprogram rendszerterve [Bendl 80b, Bendl 80c].

A forditóprogram tervezésében a fő nehézséget az okozta, hogy az IBM/ESZR architekturában nem áll rendelkezésre indirekt cimzési mechanizmus, amivel a PROLOG-ból generált kód lényegesen egyszerüsithető lenne /a létező PROLOG forditó [xWarren,77] a DEC-10 gépre generál kódot, amelyen van indirekt cimzés/. Emellett a forditóprogramban az MPROLOG-nak a PROLOG-on tulmenő szolgáltatásait is biztositani kell, ami az absztrakt, gépfüggetlen PROLOG-gépi-kód bővitését is igényelte. A terv alapján a létező MPROLOG-rendszerből, a beépitett eljárások moduljait felhasználva viszonylag rövid idő alatt elkészithető az MPROLOG forditóprogram.

5. Tapasztalatok, következtetések

A PROLOG fejlesztések és alkalmazások sikerét alapvetően a PROLOG nyelv jellegzetességeinek, nagyon magas szintü lehetőségeinek /lásd 1.3/ tulajdonitjuk. Ezeken tulmenően számos kisebb jelentőségü tényező együttes hatása játszott közre a PROLOG magyarországi sikeréhez:

- A PROLOG implementáció elég hatékony volt ahhoz, hogy az alkalmazások beinduljanak; az implementáció több üzemszerü alkalmazás esetén is megfelelő végrehajtási időket tett lehetővé.

- Két alkalmazási területen /épitészet és gyógyszerkutatás/ még 1975-ben /a megvalósitás évében/ sikeres alkalmazások fejlesztésére került sor; ezek alapul szolgálhattak a további hasonló alkalmazásokhoz.

- A PROLOG alkalmazásban résztvevők többségének kevés algoritmikus programozási gyakorlata volt; ez megkönnyitette számukra, hogy a PROLOG-ban való leiró jellegü, nem-algoritmikus programozás szemléletét hamar elsajátitsák.

- A PROLOG-nak a SIEMENS 7.755 számitógépen való installálásánál a nagy virtuális memória és az interaktiv környezet nagy lendületet adott az eddigi alkalmazásoknak, és uj alkalmazási területeket nyitott meg. /Ezek közül kiemelendő a software területe: COBOL programgenerátorok, INTEL mikroprogrambeültetés. A T-PROLOG és az LDM PROLOG-alapu rendszerek kifejlesztése is itt történt./

A PROLOG implementáció hat éves üzemeltetése során számos probléma és hiányosság is felmerült; ezek nagy részét a rendszer

karbantartása, fejlesztése során sikerült megoldani. Az alábbiakban azokat a problémákat soroljuk fel, amelyek a hazai PROLOG interpreterekben nem kerültek megoldásra:

- A PROLOG viszonylag nagy tárigénye gondokat okozott a kisebb installációkban /az MPROLOG tárgazdálkodása lényegesen hatékonyabb a PROLOG-énál/.

- Több alkalmazásban felmerült algoritmikus programrészek PROLOG-ba való beiktatásának igénye. Az esetek egy részében a problémát ugy oldották meg, hogy a PROLOG programok a FORTRAN programokhoz file-okon keresztül kapcsolódtak /pl. 3.1.1/. Az MPROLOG továbbfejlesztése során /4.3/ ez a probléma is teljesen kiküszöbölhető.

- Nagyméretü adatbázisok kezelésének igénye szintén több esetben felmerült. Egyes alkalmazásoknál /pl. 3.1.2/ a feladatot az operációs rendszer szerkesztőprogramjának felhasználásával sikerült megoldani. /A SIEMENS installációban lehetőség van a rendszer szerkesztőprogramjának PROLOG-ból való behivására/.

- A PROLOG implementáció egyszerü visszalépési mechanizmusa gondokat okozott egyes, kombinatorikus keresést igénylő alkalmazásoknál /pl. 3.4.3/. A nehézségeket speciális visszalépési mechanizmus PROLOG-ban való beprogramozása oldotta meg.

Nem hagyhatjuk ki az áttekintésből a szubjektiv tényezők megemlitését. A legfontosabb ezek közül az, hogy a NIM IGÜSZI-ben - valamint a fejlesztésbe a későbbiekben bekapcsolódó SZKI-ban - a fejlesztőgárda megfelelő elméleti és gyakorlati tapasztalatokkal, széleskörü bel- és külföldi szakmai kapcsolatokkal rendelkezett.

Fontos még kiemelnünk azt a tényt, hogy végig igen szoros és segitőkész volt a közremüködés a PROLOG fejlesztők és alkalmazók között. Az első PROLOG interpreter megjelenése után szinte azonnal megjelentek az alkalmazók, akiknek észrevételeire még a kisérleti üzem során ujabb és ujabb továbbfejlesztéssel reagáltak a fejlesztők. Reméljük, hogy a következő években a további PROLOG-fejlesztések kivitelezése, valamint azok alkalmazásba vitele is hasonló kedvező körülmények között történhet majd meg.

Irodalomjegyzék

*Andréka, 73 — Andréka Hajnal, Balogh Kálmán, Emődy Zoltán, Lábadi Katalin, Náray Miklós, Németi István: Automatikus programozó és programhelyesség-bizonyitó programok. NIM IGÜSZI tanulmány a KSH OSZI számára, 1973.

*Balogh, 75a — Balogh Kálmán, Lábadi Katalin: A matematikai logika software-alkalmazásai. Programozási Rendszerek'75 Konferencia kiadványa, Szeged, 26-44.old. 1975.

*Balogh, 75b — Balogh Kálmán: Szemantikus programellenőrző rendszer specifikációja. NIM IGÜSZI tanulmány a KSH-OSZI számára, 1975.

*Battani, 73 — G.Battani, H.Meloni: Interpreter du language de programation PROLOG. Groupe de l'Intelligence Artificielle U.E.R. de Luminy Universite d'Aix Marseille, 1973.

*Bedő, 79 — Bedő Árpád, Herényi István, Langer Tamás, Szeredi Péter: Programkészitési rendszerek /Programozás CDL-ben/. Közgazdasági- és Jogi Könyvkiadó, 1979.

*Clark, 80 — K.L. Clark, F.G.McCabe: IC-PROLOG - language features. Rreprinsts of Logic Programming Workshop, 1980. Debrecen, pp. 45-52.

xCoelho, 80 — H.Coelho, J.C. Cotta, L.M.Pereira: How to Solve it with PROLOG. Laboratório Nacional de Engenharia Civil, 2nd edition. Lisboa, 1980.

xColmerauer,72 — A.Colmerauer: Un systeme de communication homme-machine en francais. Groupe de l'Intelligence Artificielle U.E.R. de Luminy Universite d'Aix Marseille, 1972.

xHorváth, 76 — Horváth Katalin, Laufer Tamás, Lábadi Katalin, Márkusz Zsuzsanna: A QALl általános kérdés-válasz rendszer elmélete, nyelvei és vázlatos specifikációja. NIM IGÜSZI tanulmány a KSH OSZI számára, 1976.

xLPW, 80 — Preprints of Logic Programming Workshop, 14-16 July 1980. Debrecen /ed. S-A.Tärnlund/.

xWarren,77 — Warren, D.H.D.: Implementing PROLOG - compiling predicate logic programs. Dept. of Artificial Intellingence, Univ. of Edinburgh, 1977. Research Reports 39 and 40.

Bibliográfia

A következő jegyzék hazai szerzők PROLOG-témáju dolgozatait tartalmazza:

Andréka, 76 — H.Andréka, I.Németi: The Generalised Completeness of Horn Predicate Logic as a Programming Language. DAI Report, No.21., Univ. of Edinburgh, 1976.

Balogh, 77 — Balogh Kálmán, Futó Iván, Lábadi Katalin: PROLOG programellenőrző rendszer dokumentációja. NIM IGÜSZI tanulmány a KSH OSZI részére, 1977.

Balogh, 78a — Kálmán Balogh: On an Interactive Program Verifier for PROLOG Programs. To appear in Proc. of Coll. on Math. Logic in Programming, Salgótarján, North Holland Publ. Comp. 1978.

Balogh, 78b — Balogh Kálmán, Visnyovszky József: Esettanulmány. A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére. I.kötet. NIM IGÜSZI tanulmány. SOFTTECH D21, SZÁMKI, 1978.

Balogh, 78c — Balogh Kálmán, Szeredi Péter: Tervezés meta-nyelven. A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére. VI. Kötet. NIM IGÜSZI tanulmány. SOFTTECH D27. SZÁMKI, 1978.

Balogh, 79a — Balogh Kálmán: Egy logikai módszer programok szemantikus tulajdonságainak bizonyitására. Egyetemi doktori disszertáció, 1979.

Balogh, 79b — Balogh Kálmán, Sántáné-Tóth Edit, Szeredi Péter: Programtervezés logikai alapokon. NJSzT Első Országos Kongresszusának kiadványa. Szeged, 36-45. old. 1979.

Balogh, 80. — Balogh Kálmán, Farkas Zsuzsa, Sántáné-Tóth Edit, Szeredi Péter: Az LDM1-rendszer /Nagyvonalu rendszerterv/. SZKI és NIM IGÜSZI tanulmány a SZÁMKI részére, 1980.

Bán, 79 — Bán Péter, Kőhegyi János, Suhai György, Veszprémi Anna, Zsakó László: ANSWER információs rendszerének néhány kérdése. ELTE TTK tanulmány. SOFTTECH D38, SZÁMKI, 1979.

Bendl, 78 — Bendl Judit, Varga Kálmán, Kósa Márton, Balogh Kálmán: Moduláris PROLOG interpreterének specifikációja /Nagyvonalu rendszerterv/. NIM IGÜSZI tanulmány. SOFTTECH D20, SZÁMKI, 1978.

Bendl, 79a — Bendl Judit, Lugosi György, Márkus Zsuzsanna: Interaktiv levegőtisztaság-védelmi információellátó rendszer. Információ Elektronika, XIV. évf., 1.szám, 55-58 old. 1979.

Bendl, 79b — Bendl Judit, Boda Jánosné, Bogdánfy Géza, Kósa Márton, Naszvadi László, Visnyovszky József: A MRPOLOG rendszer felhasználói dokumentációja. NIM IGÜSZI tanulmány a SZÁMKI részére, 1979.

Bendl, 80a — J.Bendl, P.Köves, P.Szeredi: The MPROLOG system. Preprints of Logic Programming Workshop, 1980, Debrecen. pp. 201-210.

Bendl, 80b — Bendl Judit, Kósa Márton, Szeredi Péter: Az MPROLOG forditóprogram nagyvonalu rendszerterve. NIM IGÜSZI és SZKI tanulmány a SZÁMKI részére, 1980.

Bendl, 80c — Bendl Judit, Kósa Márton, Szeredi Péter: Az MRPOLOG forditóprogram rendszerterve. NIM IGÜSZI tanulmány, 1980. /Megjelenés alatt a SOFTTECH sorozatban./

Csuri, 80 — Csuri Ottó: A PROLOG alkalmazása müszaki rendszerek vizsgálatára. ÉTI tanulmány, 1980.

Darvas, 76a — Darvas Ferenc, Futó Iván, Szeredi Péter: Automatikus gyógyszerkölcsönhatás-kiszürő program. "Számitástechnikai és kibernetikai módszerek alkalmazása az orvostudományban és a biológiában" kollokvium kiadványa /szerk. Muszka Dániel/, Szeged. 413-422. old. 1976.

Darvas, 76b — Darvas Ferenc, Futó Iván, Kardos János, Kovács László:Programrendszer szerves vegyületek karcinogén hatásának előrebecslésére. NIM IGÜSZI tanulmány a KSH OSZI számára, 1976.

Darvas, 78a — Darvás Ferenc, Futó Iván, Szeredi Péter: Some Application of Theorem Proving Based on Machine Intelligence in QSAR: Automatic Calculation of Molecular Properties and Automatic Interpretation of Qualitative Structure-Activity Relationships. Proceeding of the Symposium on Chemical Structure-Biological Activity: Quantitative Approaches, Suhl, GDR, pp. 251-257. Akademie Verlag, Berlin, 1978.

Darvas, 78b — Darvas Ferenc, Futó Iván, Szeredi János, Bendl Judit, Köves Péter: PROLOG alapu gyógyszertervezési rendszer. Programozási Rendszerek '78 konferencia kiadványa. Szeged, 1978.

Darvas, 78c — Darvas Ferenc, Futó Iván, Szeredi Péter: A Logic-Based Program System for Predicting Drug Interactions. International Journal of Biomedical Computing, Vol.9. pp. 250-, 1978..

Darvas, 78d — Darvas Ferenc: A biológiai hatás és a kémiai szerkezet közötti összefüggés vizsgálata számitógéppel. Kémiai Közlemények, 50.kötet, 97-116. old. 1978.

Darvas, 79a — Darvas Ferenc, Szeredi János, Futó Iván, Rédei János: Egy logikai alapu kémiai információkezelő rendszer: elméleti megfontolások és gyakorlati tapasztalatok. NJSzT Első Országos Kongresszusának kiadványa, Szeged, 92-96.old. 1979.

Darvas, 79b — Darvas Ferenc, Futó Iván, Szeredi Péter: Expected Interactions of Spirololactions: Predictions by Computer. Proceedings of the Conference on the Pathogenesis of Hyperaldosteronism /Edited by E.Gláz/. pp. 219-220. 1979.

Darvas, 80a — Darvas Ferenc, Lopata Antal, Mátrai György: A specific QSAR model for peptides. "Quantitative Structure-Activity Analysis" c.könyvben /Szerk.: Darvas F./ Akadémiai Kiadó, Budapest, 1980.

Darvas, 80b — F.Darvas: Logic programming in chemical information handling and drug design. Preprints of Logic Programming Workshop, 1980, Debrecen, p.261.

Farkas, 78 — Farkas Zsuzsa, Sántáné-Tóth Edit: Az ANSWER file-kezelő rendszerének PROLOG nyelvü terve. A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére, II.kötet. SZKI tanulmány. SOFTTECH D22, SZÁMKI,1978.

Fidrich, 80a — Fidrich Ilona: Programdokumentáló rendszer - Nyelvleirás. SZKI dokumentum, 1980.

Fidrich, 80b — Fidrich Ilona: Programdokumentáló rendszer - Felhasználói kézikönyv. SZKI dokumentum, 1980.

Futó, 77a — Futó Iván, Darvas Ferenc, Cholnoky Eszter: Practical Applications of an AI Language /PROLOG/. II. Magyar Számitástudományi Konferencia, Budapest, 338-399.old. 1977.

Futó, 77b — Futó,Iván, Szeredi Péter: Mesterséges Intelligencia Nyelvek - a PROLOG nyelv. Információ Elektronika, XII.évf. 2-3.szám, 108-113. ill. 146-152.old. 1977.

Futó, 78a — Futó Iván, Darvas Ferenc, Szeredi Péter: The Application of PROLOG to the Development of QA and DBM Systems. In Logic and Data Bases /ed.H.Gallaire and J.Minker/, pp. 347-376. Plenum Press, New York and London, 1978.

Futó, 78b — Futó Iván, Szeredi János, Rédey János: Egy párhuzamos programozási lehetőségekkel is rendelkező ujelvü nyelv. SZKI tanulmány, 1978.

Futó, 78c — I.Futó, F.Darvas, P.Szeredi, J.Szeredi, P.Köves: Biodesign, a Logic-based System for Drug Design. To appear in Proc. of Coll. on Math. Logic in Programming, Salgótarján, 1978. North Holland Publ. Comp.

Futó, 79a — Futó Iván, Szeredi János, Rédei János: PAPLAN felhasználói kézikönyv. SZKI tanulmány, 1979.

Futó, 79b — Futó Iván, Szeredi János, Rédei János: A PROPHET ujelvü nyelv és alkalmazása. NJSzT Első Országos Kongresszusának kiadványa, Szeged, 146-154.old. 1979.

Futó, 79c — Futó Iván, Szeredi János, Rédei János: A PROPHET ujelvü nyelv. SZKI tanulmány, 1979.

Futó, 79d — Futó Iván és társai. COLAMI szabvány szerinti COBOL programok generálására szolgáló programrendszer. Alkalmazói leirás. SZKI tanulmány, 1979.

Futó, 80a — I.Futó, J.Szeredi: PAPLAN - an AI language for parallel problem solving. Proc. of International Conf. on Artificial Intelligence an Information-Control Systems of Robots, Smolenice Castle, Czechoslovakia, pp. 74-76. 1980.

Futó, 80b — I.Futó, J.Szeredi, K.Szenes: A modelling tool based on mathematical logic - T-PROLOG. To appear in Computational Linguistics and Computer Languages.

Futó, 80c — I.Futó, J.Szeredi, E.Baráth, P.Szabó: Using T-PROLOG for a long-range regional planning problem. Preprints of Logic Programming Workshop, Debrecen, pp.172-76. 1980.

Gerő, 80 — Gerő Péter, Halmay Dénesné: Computer aided supervision as a training of programmers. SZÁMOK report, 1980.

Garami, 78 — Garami Péter, Szeredi János: INTEL 3000 mikroprogram beültető program. A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére, V.kötet, SZKI tanulmány. SOFTTECH D24, SZÁMKI, 1978.

Herényi, 78 — Herényi István, Futó Iván: Modul-könyvtár kezelő a tervező rendszerhez. RPG-elemző tervezése PROLOG nyelven. A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére, III. és IV. kötet. NIM IGÜSZI tanulmány. SOFTTECH D23, SZÁMKI, 1978.

Holnapy, 79 — Holnapy Dezső: Az automatizált müszaki tervezés matematikai alapjai - A rendszerépitéssel izomorf algebrai struktura /Feltáró tanulmány/. ÉTI tanulmány, 1979.

Kaposi, 79a — A.A.Kaposi, Z.Márkusz: PRIMLOG the Case for Augmented PROLOG Programming. Proceedings of Informatica '79, Bled, 1979.

Kaposi, 79b — A.A.Kaposi, Z.Márkusz: Introduction of a Complexity Measure for Control of Design Error in Logic-based CAD Programs. Proceedings of CAD'80, Brighton, 1980.

Kiss V, 78 — Kiss Viktória, Simor Gábor: Egy architektura tervezési környezet előzetes specifikációja és az abban alkalmazható programsegédletek vizsgálata. SZKI tanulmány a SZTAKI részére, 1978.

Kiss V, 79 — Kiss Viktória, Simor Gábor: Kiértékelő szimulátor /DELBOLSIM/ magasabbszintü architektura tervezéséhez és kisérleti ellenőrzéséhez - ismertetés tervezők számára. SZKI tanulmány a SZTAKI részére, 1979.

Kiss Z, 79 — Kiss Zoltán, Prószéky Gábor, Tóth Lajos: Magyar nyelvü szövegek számitógépes morfológiai vizsgálata. Tanulmány a SZÁMKI részére, készült az MTA Nyelvtudományi Intézetében. SOFTTECH D41, SZÁMKI, 1979. /Angol nyelvü átdolgozott változata megjelenés alatt Computational Linguistics and Computer Languages-ben/.

Kőfalusi, 79 — Kőfalusi Viktor, Bartha Ferenc: A PROLOG alkalmazási és bővitési lehetőségeiről: állapottér-halmazokon alapuló PROLOG-alkalmazások. SOFTTECH D42, SZÁMKI, 1979.

Kőfalusi, 79a — Kőfalusi Viktor: Egyszerüsités matematikai strukturákban. SOFTTECH D42, SZÁMKI, 1979. 12-86.old.

Kőfalusi, 79b — Kőfalusi Viktor: Adott valós, nagybonyolultságu, többváltozós, analitikus függvény első n formális deriváltját előállitó PROLOG program, annak alkalmazásai, különös tekintettel a formális hatványsorba fejtésre. SOFTTECH D42, SZÁMKI, 1979. 104-127.old.

Kőfalusi, 79c — Kőfalusi Viktor, Bartha Ferenc: Ligandkötő rendszerek numerikus analizise. Tanulmány készült az MTA SZBK Enzimológiai Intézetében. SOFTTECH D42, SZÁMKI, 1979. 128-132.old.

Kőfalusi, 80 — V.Kőfalusi: The State-Space Set and its Applications in PROLOG /To appear in Computational Linguistics and Computer Languages/.

Köves, 78 — Köves Péter: BS2000 PROLOG felhasználói kézikönyv, V2.4. SZKI tanulmány a SZÁMKI részére, 1978.

Köves, 79 — Köves Péter: MRPOLOG rendszer hibakereső és nyomkövető alrendszerének előzetes felhasználói leirása. SZKI tanulmány. SOFTTECH D32. SZÁMKI, 1979.

Láng, 78 — Láng Oszkárné: Adatfeldolgozó ANSI-COBOL programok generálása PROLOG nyelven. Programozási Rendszerek '78 konferencia kiadványa, Szeged, 364-368.old. 1978.

Laufer, 79 — Laufer Tamás: DOS PROLOG Felhasználói Kézikönyv. Tanulmány a SZÁMKI részére, készült a pécsi Pollack Mihály Müszaki Főiskolán, 1979-ben.

Márkusz, 77a — Z.Márkusz: How to Design Variants of Flats Using Programming Language PROLOG, Based on Mathematical Logic. Proceedings of IFIP '77, Toronto, 1977.

Márkusz, 77b — Márkusz Zsuzsanna: A PROLOG programozási nyelv alkalmazása paneles lakóépületek építészeti tervezési feladatainak megoldására. Információ Elektronika, XII.évf. 3.szám, 124-230.old. 1977.

Márkusz, 80a — Márkusz Zsuzsanna: A PROLOG programozási nyelv alkalmazása többszintes lakóépület tervezéséhez. Információ Elektronika, XV.évf.5.szám, 265-263.old. 1980.

Márkusz, 80b — Z.Márkusz: An application of PROLOG in designing many storied dwelling houses. Preprints of Logic Programming Workshop, Debrecen. pp. 249-260. 1980.

Mátrai, 79 — Mátrai György: PROLOG program enzimszekvenciák hasonló szubstrukturáinak megkeresésére. Tanulmány a SZÁMKI részére, készült az MTA SZBK Enzimológiai Intézetében, 1979-ben.

Mátrai, 80a — Mátrai György, Darvas Ferenc, Keleti Tamás: Homologous partial seguences in dehidrogenases. /Submitted to Int.J. of Peptide Protein Research/.1980.

Mátrai, 80b — G.Mátrai: Primary structure activity of dehidrogenases. /To appear in J. of Mol.Bio. No.5., 1980./

Sántáné-Tóth, 79 — Sántáné-Tóth Edit: A hazai PROLOG-alkalmazások helyzete 1979-ben. SZKI tanulmány. SOFTTECH D40, SZÁMKI, 1979.

Sántáné-Tóth, 80 — E.Sántáné-Tóth, P.Szeredi: PROLOG applications in Hungary. Preprints of Logic Programming Workshop, Debrecen. pp. 177-189. 1980.

Szeredi J, 78 — Szeredi János: A matematikai logika alkalmazása a számitástudományban. ELTE szakdolgozat, 1978.

Szeredi P, 75a — Szeredi Péter: Egy logikai alapu magasszintü programozási nyelv. NJSzT Rendszerprogramozói Konferencia '75 kiadványa, Szeged. 191-209. old. 1975.

Szeredi P, 75b — Szeredi Péter: PROLOG interpreter dokumentációja. NIM IGÜSZI tanulmány, 1975.

Szeredi P, 76 — Szeredi Péter, Futó Iván, Lábadi Katalin: A PROLOG továbbfejlesztésének dokumentációja. NIM IGÜSZI tanulmány, 1976.

Szeredi P, 77a — Szeredi Péter, Futó Iván: PROLOG kézikönyv. SZÁMOLÓGÉP, VII.évf. 3-4.szám, 5-130.old. 1977. /Orosz nyelvü forditása készült 1979-ben/

Szeredi P, 77b — Szeredi Péter: PROLOG - A Very High Level Language Based on Predicate Logic. Preprints of Second Hungarian Computer Science Conference, Budapest. pp. 853-856. 1977.

Szeredi P, 79 — Futó Iván, Lábadi Katalin, Szeredi Péter, Balogh Kálmán /Szerkesztette: Szeredi Péter/: A PROLOG nyelv megvalósitási módszereiről és elméleti alapjairól. NIM IGÜSZI és SZKI tanulmány. SOFTTECH D34, SZÁMKI, 1979.

Szeredi P, 80 — P.Szeredi, K.Balogh, E.Sántáné-Tóth, Zs.Farkas: LDM - a Logic Based Software Development Method. Preprints of Logic Programming Workshop Debrecen. pp.172-176. 1980.

Tóth, 74 — Tóth Péter: Ujelvü programozási nyelvek összehasonlitó vizsgálata. NIM IGÜSZI tanulmány a KSH OSZI részére, 1974.