MIKROPROLOG

MATJAŽ GAMS*, TATJANA ZRIMEC**
*INSTITUT „JOŽEF STEFAN, LJUBLJANA"
**FAKULTETA ZA ELEKTROTEHNIKO, LJUBLJANA

UDK : 681.3.06:519.682

Podana je primerjava mikroprologa s prologom in izkušnje pri poučevanju. Mikroprolog hitro postaja ena najbolj razširjenih modernih variant prologa, zato je podana ocena sprememb in izboljšav glede na bolj ustaljene inačice. Poglavitni namen mikroprologa pa je vzgoja, zato so posebej analizirane vzgojne poante tega jezika, zlasti izkušnje pridobljene s poučevanjem najmlajših učencev in pri izobraževanju odraslih.

A comparison is given between MICRO-PROLOG and PROLOG and especially experiences with teaching MICRO-PROLOG. MICRO-PROLOG is one of the modern wide spread variants of PROLOG so it was interesting to evaluate improvements and changes according to more standard PROLOGs. Since the main purpose of MICRO-PROLOG is education special care was devoted to experiences with teaching youngest pupils and adults thus evaluating full spectrum of learning characteristics.

1. Uvod - zgodovina razvoja

Idejna zasnova logike kot programskega jezika se pojavlja že okrog leta 1970 /1,2/, vendar pot od ideje do realizacije ni bila enostavna in premočrtna. Že ob prvih implementacijah pa se je pojavilo prvo veliko razpotje. Ena prvih implementacij je bil PLANNER (Hewitt, M.I.T.), ki pa je po velikih in neizpolnjenih pričakovanjih kmalu neslavno propadel. Verjetno je to eden poglavitnih razlogov, da v ZDA še danes prevladuje lisp. Druga učinkovitejša in enostavnejša implementacija je bil PROLOG (PROgramming in LOGic) (Colmerauer, Marseille,1972). Ta veja se je razširila v množico bolj ali manj podobnih inačic. Mikroprolog je ena novejših inačic in verjetno tudi najbolj razširjena, saj obstaja v nekoliko prilagojeni verziji celo za hišne računalnike tipa sinclair ZX ali commodore 64. Prav tako ga dobimo na raznih operacijskih sistemih, recimo CPM/80, CPM/86 ali MSDOS/PCDOS /3/. Med avtorji mikroprologa omenimo predvsem Kowalskega, Ennalsa in McCaba /3,4,5/. Posebej zanimiv je Robert Kowalski,saj je položil temelje logičnega programiranja /1,2/. F.G.McCabe je pravzaprav glavni implementator mikroprologa, J.R. Ennals pa se je ukvarjal predvsem z uporabo mikroprologa v pedagoške namene, v okviru projekta "Logic as a Computer Language for Children". Večina dela je potekala na Imperial College v Veliki Britaniji od leta 1980 dalje.

Prolog kot implementacija logičnega programiranja je osnova japonske 5. generacije računalnikov in poleg Japonske doživlja velik vzpon predvsem v Evropi. Prolog je precej razširjen zlasti v Sloveniji in tudi drugod v Jugoslaviji. Veliko zaslug pri uvajanju tega jezika gre na račun I. Bratka /6,7/. Prolog je že nekaj let redni učni predmet na katedri za računalništvo in informatiko Fakultete za elektrotehniko Ljubljana, večino prevajalnikov za prolog pa smo nakupili ali dobili preko Instituta Jožef Stefan, kjer dokončujemo prevajalnik za prolog v pascalu.

2. Nekaj enostavnih primerov

V nasprotju s klasičnim programiranjem, ki zahteva "strog" in dobro premišljen pristop ter točno določen potek izvajanja /8/, omogoča mikroprolog programiranje vzorčno vodenih sistemov (pattern-directed systems). Taki sistemi temeljijo na samostojnih kosih informacij (pravila ali dejstva), ki veljajo za določeno problemsko področje. Interpreter sam izvaja pravila tako, da primerja, če se med podatki v podatkovni bazi pojavi ustrezen vzorec. Ravno ta zamisel, da lahko v sistem dodajamo (ali pa brišemo ali spreminjamo) pravila ali dejstva neodvisno od ostalega sistema, močno olajša programiranje še zlasti za najmlajše učence, saj je ob pravilnosti vseh vloženih pravil zajamčeno tudi pravilno delovanje celotnega sistema.

V mikroprologu običajno programiramo tako, da
1. definiramo dejstva, objekte in relacije med njimi
2. definiramo pravila, s katerimi določimo zakonitosti problemskega prostora
3. postavljamo razna vprašanja o vpisanih dejstvih in pravilih ter zaključkih sestavljenih pravil.

Tako način programiranja kot komunikacija sama ste bliže opisovanju v naravnem jeziku kot pri ostalih programskih jezikih. Na nekaj enostavnih uvodnih primerih si oglejmo izražanje v slovenščini, prologu in mikroprologu, oziroma

dodatku SIMPLE, ki ga dobimo hkrati s kaseto z mikroprologom za mikroračunalnik sinclair ZX.

```
Slovenščina: Jaka je oče od Andreja.
Prolog     : oce(jaka, andrej).
Mikroprolog: Jaka oce-od Andrej
       ali oce(Jaka Andrej)
       ali oce(jaka andrej)
```

Komentar: Pri prologu se konstante začnejo z malo začetnico, zato pišemo npr. "andrej". V mikroprologu lahko pišemo konstante z veliko ali malo. V *osnovnem prologu imamo prefiksno* notacijo, v mikroprologu (programu SIMPLE - glej dodatek 1) pa lahko izbiramo med prefiksno in infiksno notacijo za enega ali dva argumenta. Za več argumentov lahko izbiramo med prefiksno notacijo in infiksno z več argumenti združenimi v seznam.

```
Slovenščina: Janez ljubi Majdo.
Prolog     : ljubi(janez, majda).
Mikroprolog: Janez ljubi Majda
```

Komentar: V obeh prologih razumljivo ne moremo sklanjati, spregati itd., še zlasti ne v slovenščini. Zato moramo povsod pisati popolnoma enake oblike besed.

```
Slovenščina: Janez prodaja krompir.
             Janez je zelje.
             Janez prodaja paradiznik.
             Janez je krompir.
Prolog     : prodaja(janez, krompir).
             je(janez, zelje).
             prodaja(janez, paradiznik).
             je(janez, krompir).
Mikroprolog: Janez prodaja krompir
             Janez je zelje
             Janez prodaja paradiznik
             Janez je krompir

Slovenščina: Ali Janez je in prodaja zelje?
Prolog     : je(janez, zelje),
             prodaja(janez, zelje).
Odgovor    : YES
Mikroprolog: is( Janez je zelje and
                 Janez prodaja zelje)
Odgovor    : YES
```

Komentar: Konjunkcije (AND) delamo v prologu z vejicami, v SIMPLu z "and". Disjunkcije (OR) običajno delamo z več samostojnimi stavki.

```
Slovenščina: Kdo je in prodaja krompir?
Prolog     : je(X, krompir),
             prodaja(X, krompir).
Odgovor    : X = janez
Mikroprolog: which(X: X je krompir and
                      X prodaja krompir)
Odgovor    : Janez
```

Komentar: Spremenljivke so vezane znotraj enega stavka, to pomeni, da mora X znotraj stavka imeti natanko isto vrednost, dva X-a v dveh stavkih pa nimata neposredne povezave preko imena. Spremenljivke v mikroprologu so x, X, y, Y, z, Z, lahko pa jim dodamo še številko, npr.: x11.

```
Slovenščina: Janez pripoveduje:" Moja mama je
             rodila otroka, pa mi ni ne brat
             ne sestra. Kdo je to?"
Prolog     : mati(mama, janez).
             kdo(X) :- mati(mama, X),
                       not brat(janez, X),
                       not sestra(X, janez).
Odgovor    : X = janez
Mikroprolog: moja-mama je-mati-od Janez
             which(To je X:
                   moja-mama je-mati-od X and
                   not Janez je-brat-od X and
                   not X    je-sestra-od Janez.
Odgovor    : To je Janez.
```

Komentar: Pri takem preslikovanju iz naravnega jezika v katerikoli prolog je seveda treba upoštevati, da je v prologu zelo malo predprogramiranega znanja v obliki podprogramov. Zato moramo človeško znanje (angl. *common sense*) zakodirati za vsak primer posebej, npr. v zgornjem primeru moramo definirati relaciji "brat" in "sestra", čeprav program tokrat pravilno deluje tudi brez tega. V teh primerih smo prvič srečali negacijo - konstrukt "not".

3. Primerjava med prologom in mikroprologom

Eno zanimivih vprašanj o mikroprologu (programu SIMPLE) je, v kolikšni meri so dodatki in spremembe boljše kot v standardnih verzijah prologa. To bi nam povedalo, v kolikšni meri lahko pričakujemo nadaljne spremembe v razvoju logičnega programiranja. Odgovor na to je precej opurtunističen, saj spremembe le malenkostno spreminjajo osnovni koncept. Določeni dodatki kot možnost infiksnega zapisa so verjetno smiselni tudi za resno programiranje, za učenje pa so po mnenju avtorjev izredno koristni. Drugi dodatki kot možnost pisanja konstant z veliko začetnico so ugodni, vendar potegnejo za seboj težave pri pisanju spremenljivk (ni mnemoničnih spremenljivk - glej /8/). Tudi pri pisanju numeričnih izrazov na prvi pogled pridobimo v obrnljivosti in imamo tako eno izjemo manj, saj lahko numerični izraz obravnavamo kot običajno relacijo in lahko iščemo katerokoli spremenljivko. V običajnih prologih to ni mogoče, saj lahko računamo le v eno smer, npr. Y is X + 5, v mikroprologu pa bi zapisali SUM(X 5 Y) in bi lahko računali X npr. s SUM(X 5 7) ali Y s SUM(2 5 Y). Nekatere druge spremembe kot pisanje presledkov namesto vejic, pisanje "and" za konjunkcije itd. pa so bolj ali manj oblikovne narave.

Omenimo še nekatere dodatne lastnosti kot ukaze za delo z nizi, solidno realno aritmetiko (vsaj za jezike umetne inteligence), ukaze za delo z datotekami...

Čeprav so določene oblikovne razlike, pa lahko v mikroprologu naredimo praktično vse operacije kot v prologu, tako lahko uporabljamo tudi najmočnejše programske prijeme iz običajnega prologa celo na najmanjših mikroračunalnikih. Žal pa nekateri moduli skoraj ne pridejo v poštev na najmanjših računalnikih, recimo ukazi za sledenje izvajanju, saj na najmanjših mikroračunalnikih hitro zmanjka pomnilniškega prostora.

Tehnične karakteristike mikroprologa so prav tako med najboljšimi glede na ostale inačice. V enem stavku bi jih našteli z: *izredno majhen in hiter prevajalnik z nekaj opcijami* (osnovni mikroprolog, standardni prolog, SIMPLE...), modularna struktura, izdelana komunikacija s perifernimi napravami.

Modularno programiranje je močno izraženo. Večji program razbijemo na logično zaključene manjše dele in jih deklariramo za module. Med izvajanjem lahko izvajamo samo nekaj modulov. Ko potrebujemo kakšen modul, ki ga nimamo v centralnem pomnilniku, ga naložimo z ustrezne periferne enote (diskete, diska, ...) in ga ponovno odložimo, ko ga ne potrebujemo več. Tako lahko izvajamo precej obsežnejše programe, kot pa nam to omogoča pri mikroračunalnikih precej omejeni centralni pomnilnik.

Tudi komunikacija s perifernimi napravami je zelo učinkovita in enostavna, saj preko vmesnika RS232 dosegamo naprave od tiskalnikov do gibkih ali trdih diskov.

Literatura o mikroprologu je dokaj solidna, ponekod pa je nepotrebno prilagojena mlajšim učencem. Oglejmo si primer iz /4/ s strani 129:

```
(x y) sums-to z if SUM(x y z)
(x ! y) sums-to z if y sums-to X and SUM(X x z)
```

če dobro premislimo, bi bilo namesto prvega stavka bolje: () sums-to 0
saj je krajše, bolj univerzalno in bolj "čisto" logično razmišljanje. Take drobne ohlapnosti so nepomembne pri programiranju malih primerov, pri zahtevnejših programih pa so eden poglavitnih vzrokov za nezanesljivo in nepravilno delovanje, saj podobno kot gornji primer puščajo prostor za nepredvidene situacije (npr. napaka v primeru iskanja vsote enega elementa).

V celoti gledano pa so zbirke nalog med boljšimi.

4. Mikroprolog kot šolski jezik

Mikroprolog je zanimiv zlasti kot šolski jezik. Naštejmo osnovne značilnosti (prednosti) mikroprologa kot učnega jezika:

- perspektivnost

Zgodovina računalništva /8/ nas uči, da gre razvoj programskih jezikov v smeri približevanja ljudem. Od strojnega jezika preko zbirnega in visokih algoritmičnih jezikov pridemo do deklarativnih jezikov tipa prolog. V japonskem projektu 5. generacije računalnikov pa pride do revolucionarne spremembe, kjer je strojni jezik kar inačica proloqa, imenovan KLO (Kernel Language 0). Nad njim srečamo druge inačice, npr. ESP na nivoju makrozbirnika. Torej lahko pričakujemo, da bodo jeziki prihodnosti če že ne inačica proloqa, pa vsaj precej bolj podobni prologu kot npr. fortranu. Kakšen smisel je učenje že zdaj zastarelih jezikov (tipa fortran ali cobol), ki bodo še precej bolj zastareli (če ne cjlo izumrli), ko bodo učenci dokončali študij?

- razširjenost

Današnji pogoj za uporabnost programskega jezika je razširjenost na vseh, tudi najmanjših mikroračunalnikih. Mikroprolog je ne samo dosegljiv na hišnih računalnikih, ampak ima celo posebej prilagojene dodatke za začetnike, npr. program SIMPLE /3/.

- splošnost

V nasprotju z učenjem specializiranih področij znotraj računalništva naj bi bila osrednja tema učenja z mikrologom učenje LOGIČNEGA RAZMIŠLJANJA. Programski jeziki se menjajo, prihajajo programski paketi in sistemi, ki sploh ne zahtevajo znanja programiranja na nivoju programskih jezikov, vsako smiselno človeško opravilo pa je vezano na logiko. Z mikrologom se lahko učimo programirati kot s programskim jezikom, prav lahko pa se učimo tudi druge predmete, npr. matematiko, zgodovino itd. /4/. Ne glede na to, iz kakšnega področja so vaje, pa vedno podpirajo razvijanje logičnega mišljenja.

5. Kratek pregled učne snovi za najmlajše učence

5.1. vaje po učni snovi

Poglejmo nekaj vaj v mikroprologu (po vzoru iz /4/) najprej po učni tematiki:

MATEMATIKA:

```
x ima-povprečje y if x ima-vsoto z and
                     x dolg X and
                     deljeno(z X y)
```

Kar preberemo kot: povprečje od x je y, če je vsota elementov iz x enaka z in je x sestavljen iz X elementov in y = z/X.

Podprograme "ima-vsoto", "dolg" in "deljeno" moramo sami sprogramirati:

```
() ima-vsoto 0
(x ! y) ima-vsoto z if y ima-vsoto z1 and
                       SUM(z1 x z)
```

Komentar:
Vsota praznega seznama je 0.
Vsota seznama s prvim elementom x in ostankom seznama y je z pri čemer je y vsota z1 in z = z1 + x.

```
() dolg 0
(x ! y) dolg z if y dolg z1 and
                  SUM(z1 1 z)

deljeno(x y z) if TIMES(y z x)
```

Operacije nad množicami:

```
x presek (y z) if x isall(X: X ON y and X ON z)
x razlika (y z) if x
                   isall(X: X ON y and not X ON z)
x član X if x ON X
```

Komentar: Npr. presek: množica x je presek množic y in z, če je sestavljena iz elementov, ki so v y in z hkrati. Tu smo srečali konstrukt "isall", ki zgradi seznam iz elementov, ki ustrezajo logičnemu pogoju v oklepajih.

ZGODOVINA:

```
(začetek druge svetovne vojne) datum 1941
Informbiro datum 1948
(konec druge svetovne vojne) datum 1945
which(x se je zgodil leta 1941: x datum 1941)
(začetek druge svetovne vojne) se je zgodil
  leta 1941

which( Informbiro je bil leta X: Informbiro
  datum X)
Informbiro je bil leta 1948
```

```
Hitler je nacist
nacist hoce nadvlado
nacist hoce vojno
Učenec skuša odgovoriti na vprašanje, kaj hoče
Hitler:
which(x: Hitler hoce x)
No (more) answers
(prvo vprašanje ni rodilo sadov)

which(x: Hitler je x)
nacist
which(Hitler hoce x: nacist hoce x)
Hitler hoce nadvlado
Hitler hoce vojno
```

Z gradnjo obsežnih baz, ki vsebujejo veliko med seboj povezanih podatkov, lahko učenci povezujejo informacije med seboj in se tako učijo spoznavati povezave, ne samo gola dejstva. Poleg tega je enostavno pisati programe za simulacijo zgodovinskih dogajanj /4/. Ti programi so prav tako primerni za učenje zemljepisa, sociologije, spoznavanje narave, učenje kemije ali arheologije /9/.

UČENJE JEZIKOV

Za šalo prepišimo mikroprolog v slovenščino:

```
dodaj(x)   if add(x)
kateri(x)  if which(x)
en(x)      if one(x)
vsota(x y z) if SUM(x y z)
x element y  if x ON y
dodaj(Janez ljubi Micka)
kateri(x: x ljubi Micka)
Janez
```

Poskusimo še s prevajanjem med jeziki:

```
I          je jaz
you        je ti
sun        je sonce
see        je vidim
hollidays  je počitnice
sea        je morje
and        je in
()       angl-sl ()
(x ! X) angl-sl (y ! Y) if x je y and
                           X angl-sl Y

which(x: x angl-sl (jaz vidim sonce))
I see sun.

which(x: (see hollidays and sun and sea)
         angl-sl x)
vidim počitnice in sonce in morje
```

Komentar: zaradi bogastva slovničnih oblik v slovanskih ali germanskih jezikih takšni primerčki niso tako uspešni kot npr. v angleščini ali francoščini.

Seznam vaj iz računalniškega vrtca je v dodatku 2.

Po mnenju avtorjev je ena večjih prednosti poučevanja mikroprologa ravno v obsežni zbirki zanimivih vaj, ki so hkrati zanimive in poučne. Tako se učimo postavljati vprašanja (query) v detektivskih nalogah, npr. iščemo Jacka Razparača v podatkih o osumljencih, izbiramo miss sveta izmed kandidatk itd. Po drugi strani pa je zaporedje učne snovi v /4/ nekoliko prepočasi tempirano za sposobnejše učence, saj prepočasi pride do praktičnega programiranja.

Ena večjih pomanjkljivosti predvsem za mlajše učence je pomanjkanje specializiranih konstruktov za risanje. Tako v basicu kot v logoju večina začetnega poučevanja temelji na grafiki, v mikroprologu pa kljub podatkom iz literature nismo uspeli dobiti posebne grafike na sinclairju ZX in učnih materialov zanjo. Seveda pa lahko sami napišemo svoje grafične rutine s pomočjo izpisovanja krmilnih znakov.

6. Izkušnje s poučevanjem

V /4/ Ennals predstavlja šolske materiale za britanski projekt (največ v obliki seznama vaj). Učenje je potekalo v običajnih šolskih razredih z enim računalnikom in velikim zaslonom. V razredu je bilo 26 učencev v starosti 10-11 let. Pouk je potekal v okviru rednega pouka in je trajal 3 uri 20 minut tedensko. Učenci so imeli popoldne možnost učenja na šolskih računalnikih.

V Jugoslaviji je potekalo učenje mikroprologa na Institutu "Jožef Stefan" v okviru izobraževanja zaposlenih na Institutu in v okviru Računalniškega vrtca za učence v starosti 10-14 let /10/. Pouk štirinajstih mladih učencev je potekal v 18 učnih sklopih vsakič po dve uri in 15 minut, izobraževanje petnajstih zaposlenih pa v 8 sklopih po 2 uri. Učenje je potekalo v računalniških učilnicah z računalniki sinclair ZX z 48 K in monitorjem na vsaki mizi.

Težave pri poučevanju:
Težave pri praktičnem poučevanju so izvirale iz več vzrokov. Nekaj težav je izviralo iz opreme, tako npr. je bilo tipkanje na računalnikih sinclair z gumijastimi tipkami izredno zamudno. Pri najmlajših učencih je bilo opaziti tudi probleme z motiviranostjo, saj bi se včasih rajši igrali kakšne igrice. Pri izobraževanju računalniško podkovanih kadrov je bilo opazno razumevanje tudi na globjem proceduralnem nivoju, medtem ko so učenci v računalniškem vrtcu ostali na nivoju deklarativno-intuitivnega razumevanja rekurzije in nekako niso uspeli preskočiti tega praga.

Rezultati testiranja:
Na predlog dr. M. Ribariča (idejni pobudnik in vodja celotnega projekta računalniških vrtcev) smo poskusili ugotoviti, ali je osnovni cilj učenja z mikroprologom uspel ali ne. Ker je osnovni cilj učenje logičnega razmišljanja, smo izbrali naloge - logične uganke - iz /11/ in primerjali rezultate testov ob začetku in koncu tečaja iz mikroprologa v računalniškem vrtcu. Učenci so reševali pismene teste načeloma v skupinah po dva, tako kot je običajen način dela v vrtcu. Obakrat so imeli reševalci na voljo 25 minut. V tabelah vidimo rezultate prvega in drugega testiranja. Številke na levem robu pomenijo številke nalog iz /11/, številke na zgornjem robu zaporedno število izdelka, enice znotraj tabele pa uspešno rešene naloge.

Rezultati prvega testiranja:

	1	2	3	4	5	6	7	8
26	1	1	1		1	1	1	1
27								
28	1					1		
34						1	1	
37				1		1	1	
41								

3. izdelki niso dobili nobene točke
Skupno 14 rešenih nalog, 11 izdelkov.
Število rešenih nalog na izdelek: 14/11 = 1.27.

Rezultati drugega testiranja:

	1	2	3	4	5	6	7	8	9
29	1	1	1	1	1	1	1	1	
30		1				1		1	
32									
36			1	1	1	1	1		1
38	1	1			1			1	1
39									

Skupno 22 rešenih nalog, 9 izdelkov.
Število rešenih nalog na izdelek: 22/9 = 2.44.

Rezultati testiranja nakazujejo pozitiven vpliv učenja mikroprologa na logično razmišljanje, saj je število uspešno rešenih nalog na izdelek narastlo z 1.27 na 2.44. Vendar ti rezultati nimajo neoporečne znanstvene teže, saj je izstopilo nekaj učencev, ki je imelo največ težav, tudi zasedba dvojic ni bila enaka med prvim in drugim testiranjem. Zaradi tega imajo rezultati le informativno vrednost. V obeh testiranjih so bile izbrane približno enako težke naloge, tako da je bila ena očitno pretežka, za ostale pa smo upali, da jih bodo lahko rešili. Glede na dosežene rezultate pa so bile nekatere naloge le pretežke, zato bi bilo za večjo objektivnost rezultatov smiselno ponoviti testiranje z večjim številom lažjih nalog. Učenci so se večinoma pritoževali, da so bile drugič naloge težje kot prvič. Kljub vsemu pa testi kažejo na to, da obstaja močan pozitiven učinek učenja mikroprologa na logično razmišljanje, saj se je po vsem sodeč močno dvignilo povprečje teh učencev.

Poglejmo si število dvojic učencev, ki so rešili po 4, 3, 2 in 1 nalogo v obeh merjenjih:

Nalog	Prvič	Drugič
4	1	0
3	1	4
2	1	5
1	5	0
0	3	0

7. Diskusija

Osebne izkušnje: Pri prehodu s prologa na mikroprolog so določene težave, čez nekaj časa pa se pokaže, da je svoboda izražanja nekoliko večja v mikroprologu, oblika zapisa pa nekoliko enostavnejša. Tudi poučevanje v mikroprologu je bolj učinkovito. Kljub pogostim trditvam, da je prolog enostaven za učenje, se avtorja ne strinjata povsem s tem. Učenje prologa je predvsem učinkovito in hitro, to pomeni, da lahko učenci kmalu začnejo pisati zahtevne programe. Pred tem pa morajo narediti zahteven miselen preskok. Ta preskok je dokaj enostaven za tehnično izobražene kadre, npr. matematike ali za usmeritve, ki podpirajo logično razmišljanje. Pri drugih izobrazbah pa se pokažejo težave. Posebno rekurzija in proceduralni pomen prologa delajo veliko težav tudi računalniško izobraženim kadrom, ki so navajeni programirati v starejših jezikih brez rekurzije (cobol, fortran). Z mikroprologom je učenje potekalo opazno hitreje in bolj uspešno.

Mikroprolog kot programski jezik: S stališča profesionalnega programiranja mikroprolog ne prinaša velikih sprememb glede na ustaljene inačice kljub temu, da ima nekaj dodatkov, ki nekoliko polepšajo logično celovitost jezika. Ti dodatki so včasih celo nekoliko dolgovezni s stališča izurjenega programerja, zato pa so veliko bolj koristni za učenje programiranja, saj ne samo olajšujejo učenje, ampak omogočajo še nekoliko višji - bolj deklarativni nivo in s tem dajejo možnost večjega poudarka učenju logičnega razmišljanja. Druga misel, ki se ponuja, pa je, da je osnovni mehanizem prologa zelo močno orodje, ki ga inačice niti v peti generaciji ne bodo bistveno spremenile vsaj z logičnega stališča. Tehnično-uporabniško gledano pa je mikroprolog veliko bliže jezikom za profesionalno programiranje, saj ima izdelano komunikacijo z zunanjimi napravami, omogoča modularno programiranje, prevajalnik sam pa je med najmanjšimi in najhitrejšimi.

Mikroprolog kot učni jezik: Mikroprolog je verjetno še nekoliko primernejši za učenje glede na ostale inačice, še posebej pokaže prednosti pri poučevanju učencev z ne pretirano predizobrazbo. Celo primerjava s programskim jezikom logo /4/ pokaže kar nekaj prednosti za učence, ki so stari nad 10 let. Le za mlajše učence pod 10 let je mikroprolog pretežak. Poglavitne prednosti učenja z mikroprologom so razvijanje logičnega mišljenja, razvijanje kreativnega in samostojnega razmišljanja in učenja programerskih principov pete generacije računalnikov.

8. Literatura

1. Kowalski R.: Predicate logic as programming language, Proc. IFIP 74 Conf., North-Holland 1974

2. Kowalski R.: Algorithm = Logic + Control, CACM, Vol. 22, No.7, 572-595, 1979

3. Clark K.L., McCabe F.G., Ennals J.R.: A Micro-PROLOG Primer, Logic Programming Associates, 1983

4. Ennals R.: Beginning Micro-PROLOG, Ellis Horwood Ltd., London, 1904

5. McCabe F.G., Clark K.L., Steel B.D.: Micro-PROLOG Programmers Reference Manual, Logic Programming Associates, 1984

6. Bratko I., Gams M.: Prolog: osnove in principi strukturiranja podatkov, Informatica 4/1980, str. 40-46, 1980

7. Bratko I.: Expert Systems and Prolog, Pergamon Infotech State of the Art Report: Supercomputer Systems Technology, F. Sumners (ed.) Series 10, No. 6, Pergamon Infotech Ltd., 1982

8. Gams M.: Osnove dobrega programiranja, Cankarjeva založba, 1985

9. Ennals R.: Micro-PROLOG across the Curriculum, Collected papers 1982-1984, Research Report DoC 84/17, Imperial College, 1984

10. Gams M.: Mikroprolog, seminarski materiali, str. 50, 1985

11. Smullyan R.M.: Alica v deželi ugank, Državna založba Slovenije, Ljubljana 1984

DODATEK 1

4. Seznam osnovnih ukazov za delo s programom "SIMPLE"

Opomba: Nekateri ukazi so vezani na sinclair ZX:

"caps shift" "Z" "break space" istočasno (prekinitev izvajanja)
"caps shift" "a" istočasno (no scrool)

accept -za vpisovanje več stavkov z isto relacijo.
Primer:
```
accept ima-rad
ima-rad.(Lojze Mojca)
ima-rad.(bbb ccc)
ima-rad.end
```

add -za dodajanje stavkov bazo podatkov
Primer:
```
add(Jure ima-rad Meri)
```
(doda stavek v bazo)
```
add 2 (Mira ima-rad Miha)
```
(doda stavek na tretje mesto med stavke relacije "ima-rad")

delete -zbriše stavek dane relacije z ustrezno zaporedno številko
Primer:
```
delete ima-rad 1
```

edit -editiranje in popravljanje stavkov programa.
Splošna oblika:
```
edit ime-relacije zaporedna-številka-stavka
```
Primer:
```
edit ima-rad 1
1 (Lojze ima-rad Mojca)
```
Zdaj se lahko premikamo po stavku s tipkami/puščicami in ga sproti popravljamo.

kill -uniči vse stavke za dano relacijo ali cel program.
Primer:
```
kill ima-rad
```
(zbriše vse stavke relacije ima-rad)
```
kill all
```
(zbriše ves program)

list -izpiše vse stavke dane relacije ali cel program.
Primer:
```
list ima-rad
```

(izpiše vse stavke relacije "ima-rad")
list all
(izpiše cel program)

which -išče vse možne odgovore.
all -sinonim za "which".
one -daje samo en odgovor naenkrat.
is -preveri pravilnost in odgovarja z "YES" ali "NO".
isall -grajenje seznamov.
Primer:
which(x:x isall (y:y ima-rad Mezi))
(izpiše se seznam vseh, ki imajo radi Mezija)
not -negiranje pogojev.
load -naloži program v računalnik z kasetofona. Splošna oblika:
load ime-programa
save -shrani celotni program iz računalnika na kasetofon. Splošna oblika:
save ime-programa
reserved -za iskanje rezerviranih besed.
Primer:
which(x:x reserved)
(izpiše vse rezervirane besede)
defined -če hočemo izvedeti, katere relacije so definirane.
Primer:
which(x:x defined)
(izpiše imena vseh relacij, ki so definirane v našem programu)
ON -članstvo v seznamu. Je izpolnjen, ko je objekt član seznama.
Primer:
is(5 ON (1 2 3 4 5))
YES
CONS -doda objekt na prvo mesto v danem seznamu.
Primer:
which(x:CONS(3 (4 5 6)))
(3 4 5 6)
No (more) answers
APPEND -združi dva seznama v tretjega.
Primer:
which(x:APPEND((1 2 3 4)(5 6 7 8) x))
(1 2 3 4 5 6 7 8)
No (more) answers

DODATEK 2

Tu so zbirno navedene vaje tečaja iz mikroprologa, ki je potekal v okviru računalniškega vrtca /10/:

0. seznanjanje z računalnikom, seznanjanje z osnovnimi ukazi tipa "add", "list","accept", "delete", "kill", "edit"
1. enostavni stavki, opisovanje podatkov/stanj (ukaz "add", vaje na nivoju "Pavle ima-rad Meri", vaje na nivoju opisovanja sedežnega reda učencev v učilnici)
2. postavljanje vprašanj (ukaz "is", enostavne vaje)
3. pretvarjanje stavkov iz slovenščine v mikroprolog (trditve, vprašanja, vaje v pretvarjanju drugih jezikov)
4. uporaba spremenljivk in ukaza "which" (vaje na nivoju enostavnih družinskih relacij)
5. sestavljena vprašanja s konstruktom "is" (tvorjenje logičnih konjunkcij z uporabo konstrukta "and")
6. sestavljena vprašanja s konstruktom "which" (tvorjenje logičnih konjunkcij z uporabo konstrukta "and")
7. vaje - izpraševanje o vpisanih podatkih - kaj je nad, višje..
8. enostavna aritmetika (LESS, SUM, TIMES in uporaba teh konstruktov za računanje)
9. splošna oblika stavka "which"
10. pravila (A if B and C and ...)
11. pravila z uporabo spremenljivk (vaje, npr. detektivske naloge tipa iskanje Jacka Razparača)
12. predstavitev enostavnih pravil z diagrami/mrežami (vaje iz družinskih relacij, npr. Jože je-oče-od Lojze)
13. predstavitev pravil s spremenljivkami v obliki semantičnih mrež (vaje iz družinskih relacij, npr. x je-oče-od y if y je-sin-od x)
14. prehod na zapletene oblike pravil, ki se med seboj povezujejo (npr. stric je brat od očeta ali ...)
15. aritmetične in logične relacije (npr. kdo je višji ali starejši od koga)
16. seznami (npr. pretvarjanje iz slovenščine v mikroprolog)
17. dostop do elementov seznama (uporaba vzorcev in operatorja za ločevanje glave od seznama)
18. vaje s seznami
19. konstrukt "one"
20. vaje s seznami in bazami podatkov (npr. izbiranje miss sveta, iskanje asociacij, zamenjevanje besed ...)
21. rekurzija (vaje na nivoju družinskih relacij - npr. "prednik")
22. članstvo v seznamu (konstrukt "ON" in njegova definicija)
23. vaje s seznami (programi za dolžino seznamov, združevanje seznamov)
24. negacija (uporaba konstrukta "not")
25. enomestne relacije (npr. Nika bere)
26. grajenje seznamov (uporaba konstrukta "isall")
27. naloge s seznami (seštevanje elementov seznamov, množenje elementov seznamov)
28. nekoliko težje naloge s seznami (kupovanje najprimernejšega avtomobila , iskanje raznih povprečij)
29. grajenje verig (npr. iskanje verig potomcev v družinskih drevesih)
30. vaje (npr. potapljanje ladjic, ugibanje skritih informacij, iskanje "kdo je najvišji" itd.)
31. sistemski ukazi