A tanuló modellje

FARKAS I. JÓZSEF

Mióta a hazai oktatási intézményekben a számítógépek használata mindennaposá vált, a magyar neveléselmélet művelőinek határozott törekvése, hogy megfogalmazzák az ember-gép kapcsolat pedagógiai szempontú alaptételeit. Ma, amikor már nem vitás az informatika létjogosultsága, sajnálatos, hogy még nem született olyan tanulmány, amely a mesterséges intelligencia kezelési stratégiáját, értékelő metodikáját, alkalmazásának pedagógiai célrendszerét kidolgozza. Sürgető az igény egy tudományelméleti alapokon nyugvó elemzésre, ugyanis a magyar iskolarendszer erősen klasszifikációs metodikái és viszonylagosan merev közvetítési keretei gátakat emelnek a számítógépes tanítás-tanulás bevezetése elé. A megalapozási törekvések részeként, jelen dolgozatban megkíséreljük felvázolni azokat a rendszereket, amelyekkel az amerikai oktatáselméletek „lazább", siker- és teljesítményorientált, erőteljesen individualizált szerkezetét illeszthetjük a hazai, illetve európai megtanító típusú, adatmanipulációs készségek, és rendszerező-kategorizáló képességek kifejlesztésére épülő hierarchikus rendszerébe.

Pedagógusok örökzöld vitatémája a tudást befogadó indivíduum korrekt jellemzésének metodikája. Könyvtárnyi a téma irodalma, azonban egyetlen olyan közelítést sem találunk, amelyikkel egy általánosítható rendszert építhetnénk fel. Pedig szükség volna egy szigorúan a vonatkozó tárgyi tudás objektív birtoklását, annak alkalmazási rutinját, általánosító, mélységét mérő, formalizált segédletre. Gondoljunk a különböző szakterületek átfedéseire, az interdiszciplináris tudományok klasszifikáció-elmosó hatásaira és vonzataira. Ilyen értelmezéssel a minősítés egységesítését, a tetszőleges tárgyi összemérhetőség gondolatait akár követelményként is deklarálhatjuk. Szembeszökő ez az igény a számítógépes oktatás értékelésében, az oktató-számítástechnika e legintenzívebben kutatott témájában.

Nem hat az újdonság erejével, ha egy computeres szakember a pszichológia területén keresi egy probléma megoldását. Sőt, ha fejlesztő oktatóprogramot kíván készíteni, szembetalálja magát ezzel a humán szférába sorolt tudományággal. A személyi számítógépek alkalmazása az oktatásban, egyes rész- és komplex feladatok kimunkálása, jól algoritmizálható mechanizmusok megtanítása kihívás szakembereink számára. Az alkalmazott programoknál ma már nem elegendő, ha gépi-tutor rendszerünk – az *expert module* – a szakterület valamennyi esetére felkészült tudásbázissal rendelkezik. Szükség van egy olyan szabadon pufferelhető szegmensre is, amelynek segítségével egy tetszőleges felhasználó pillanatnyi tudásáról, anyagfeldolgozási stratégiájáról, problémamegoldó technikájáról, sőt, érzelmi hozzáállásáról kapunk formulázható „jelentéseket". Az élő tanítás problémái bukkannak fel itt is, keresve a „lélektelen" gépi metodikák emberközeli interpretációs eszközeit. Behozhatatlannak tűnő hátrány azonban, hogy amíg egy pedagógus tapasztalataira támaszkodva, a fellépő konfliktushelyzetek folytonosan változó momentumait akár egyetlen mozdulattal, pillantással, személyes varázzsal oldani tudja, addig a számítógépes adatáramlás kommunikációs lehetősége kemény korlátot szab az eseti kezelésnek. Hogyan lehet mégis hatékonyan kitűzött célunk felé fordítani a diákság szellemi frisseségét – gépi intelligenciával? Milyen módszerek, rendszerek állnak rendelkezésünkre, miként tudjuk alkalmazni a kognitív tudományok törvényszerűségeit, a gondolkodáskutatás megállapításait egy számítógépes oktatóprogram elkészítésekor? Hogyan tudjuk „tetten érni" a részigazságok tévútra vezető hatóerőit, a hiányos ismeretek

implikálta helytelen konklúziók hatásmechanizmusát? Ezekre a kérdésekre keressük a választ, és próbálunk megadni egy formalizálható modellt, amelyet beépítve egy ITS tudásbázisába, mint a *tanuló modelljét* fogjuk nyilvántartani. Nem érdektelen azonban az élő tanítás értékelő mechanizmusainak adaptációs elemzése sem, hiszen az eljárás az emberi elme kategorizáló, információfeldolgozó stratégiájára, illetve annak szimulálására épül.

Alapozás

Mielőtt rátérnénk az elképzelt modell kifejtésére, röviden tekintsük át azt a humán tudásreprezentációt, amelyre az új diagnosztikai modul alapoz.

Az emberi agy pszicho-pedagógiai szempontból két különböző részre tagolható: feldolgozó mező és tároló mező. *A szenzoros regiszterek és a rövidtávú memória terminusok használata esetünkben is kézenfekvő volna, azonban használatukkal a kifejtés során feloldhatatlan ellentmondásba keverednénk. Ugyanis az említett megnevezéseket alkalmazó modellek egyelőre figyelmen kívül hagyják azt az orvos-fiziológiai tényt, mely szerint az agyban a jól elkülöníthető funkcionális részletek együttműködve, párhuzamosan dolgozva kezelik az információtömeget, továbbá nem nyújtanak kielégítő képet az egyes részletek közötti, belső adatáramlás törvényszerűségeinek és következményeinek egzakt magyarázata során. Ezt a hibát szeretnénk kiküszöbölni. Az Ebbinghaus, illetve James úttörő munkássága nyomán kialakult memóriamodellek felépítéséből az új irányvonalnak –, Larry R. Squire cikkében összefoglalt megállapításait tekintjük kiindulási alapnak.* Összegezve és tömörítve a közelmúlt ismertté vált elméleti közelítéseit, a következőket mondhatjuk: a külvilágból három ingertípus (hullám, mechanikai, vegyi természetű) impulzusai érkeznek az agy *feldolgozó-mezőnek* nevezett területére. Itt mint bio-elektromos impulzus a megfelelő sejthalmazokat *(rekeszeket)* ingerületi állapotba hozza. Az információ előszűrt, mértéke és minősége a szenzorok állapotától és a feldolgozó-mező szenzoradekvát „rekeszei"-nek súlyozottságától függ. A gerjesztett szegmens-értékek összegződnek, s ez az összeg (amelynek neve legyen *címke*) adja meg azt az agyterületi lokalizációt, ahol az érkező input kezeléséhez szükséges adatok tárolódnak. Ezek a területek tehát akkor aktiválódnak, ha a címke gerjesztési tartománya megközelíti a „megszólított" részlet rezonancia-frekvenciáját.

Tegyük fel, hogy minden összetartozó információ, illetve az egyidőben érzékelhető adatok halmazának szenzorok által kódolt elemei szigorúan megadható határú „modulokban" tárolódnak. A modulokat a különböző bio-elektromos állapotú sejthalmazokból csomópontok *(nodok)* sokasága építi fel, s ezek állapotösszege a modul címkéje, azaz a címzési hullámhossza. Az egyes nodok hordozzák azokat a kódsorozatokat, amelyeket az input kiértékelésekor a szenzorok generáltak a nekik megfelelő rekeszben, illetve azt az állapotot, amelyet egy-egy „frissítő hullám" pillanatában rögzíthetünk a feldolgozó-mező rekeszeiben. Egy-egy ilyen területi egységre mint tudáselemre hivatkozhatunk. Az egyes nodok közötti kapcsolatok minőségét logikai formalizmusokkal leírható hierarchikus relációk adják meg.

Minden modul több csomópontot tartalmaz, mint amennyi egy input azonosításához minimálisan kell (azaz a szükséges és elégséges gerjesztő-állapot szummája), nyilván azért, hogy a különböző kontextusokban előforduló esetek mindegyikét kezelni tudja, illetve a válaszok relevanciáját biztosítsa. *Az a kérdés, hogy az egyes nodok szummája miként adhatja így a gerjesztési frekvenciát? Válasz: a logikában és a halmazelméletben egyaránt ismert fogalom használatával, az idempotens művelet alkalmazásával magyarázható és modellezhető a jelenség!*

Megalkotandó modellünkben a különböző ingertípusok által generált címkék a fizikai valóságtól eltérő rekeszállapotokkal is kiválthatók *(képzelőerő!)*, és a kimenet csak akkor realizálódik, ha arra a szabályozó mechanizmus „engedélyt" ad.

A modellalkotás korai kísérletei

A gépi oktatás kérdésével már az ipari forradalom idején is foglalkoztak, de a programozott oktatás születési dátumának a legtöbb szerző 1954-et jelöli meg, amikor megjelent *B. F. Skinner The Science of Learning and Teaching* című írása. Az operáns tanuláselmélet kidolgozásával széles távlatok nyíltak a problémamegoldással társított önálló, alkotó gondolkodás algoritmizálására. Egy másik, ugyancsak gyakran idézett dátum: 1956 nyara, amikor a Dartmouth College-ban tartott informatikai konferencián vezető tudósok hivatalosan is deklarálták az AI *(Artificial Intelligence)* mint tudományág önállósulását. Bármelyiket is helyezzük előtérbe, a korai számítógépes oktatási rendszerek megalapozott tudományelméleti megállapításokon nyugszanak. Az első közelítések azonban, ha szigorúan vizsgáljuk, még semmilyen ellenőrizhető nyomkövetést sem alkalmaztak a tanuló munkájának nyilvántartására.

Elemezve a ráfordítás és az eredményesség mutatóit, a korai *CAI (Computer Assisted Instruction)* fejlesztők nagyon hamar rádöbbentek, hogy a gépi oktatásban addig alkalmazott parancsnyelvi kommunikációval, az egyszerű felelet-választásos kikérdezési teszteléssel, a visszajelzéseket nem kezelő technikával az elvárások nem teljesíthetők. Ezért többirányú kutatással keresték az élő tanítás lehetséges szimulálásának módjait. Mivel a számítógép kommunikációs csatornájának sávszélessége erősen korlátozott (keyboard, monitor, esetleg analóg/digitál hangátalakító az élő beszéd generálásához), szükség van egy önálló modulra, amellyel követhető a felhasználó munkája. Elsőként *J.D. Fletcher* fogalmazta meg, hogy a beérkező inputokból meg lehet határozni a tanuló jellemzőit. Megfelelő algoritmusokkal kiértékelt válaszok, illetve az inputként érkező kérdések elemzésével képet kaphatunk a felhasználó mentális szokásairól, gondolkodási struktúrájáról, tudásáról. Tőle származik a modul megnevezése is: *STUDENT MODULE*. Mielőtt a megvalósítható rendszerek áttekintését megkezdenénk, vázlatpontokba szedve fussunk végig azokon a gondolatokon, amellyel objektív szempontok alapján körvonalazhatjuk a tanuló modelljét.

A tanuló modellje

A számítógép nem felejt, nem téveszt, pontosan úgy dolgozik, ahogy azt előírták. Ezért elvárható, hogy a *Student Module*, amely felépíti a tanuló modelljét, mint gépi-tutor tartsa nyilván a kognitív diagnózison alapuló értékelések adatait a

- felhasználó(k) azonosításához szükséges elemekről
- pillanatnyi teljesítményről
- aktuális tudásszintről
- haladási ütemről
- megoldási stratégiákról
- hibákról és azok tendenciájáról
- előmenetelről és terhelhetőségről
- mentális viselkedési formákról.

A modell *didaktikai* komponensként tartalmazza egy tetszőleges felhasználó jellemzőihez adaptálva a

- tanítás tudományát
- közvetítő stratégiák specifikációit
- oktatástechnikai fogásokat
- pedagógiai elveket
- konfliktuskezelő rutinokat.

Hogyan készítsünk tehát munkatervet egy ITS számára úgy, hogy az osztályközösségben várható teljesítményekhez hasonlítani lehessen a számítógéppel dolgozó tanuló tudásszintjét?

1) Állítsunk fel tervezési stratégiát, amelyben szerepel a megtanítandó anyagrész feldolgozásához szükséges

– feltételek pontos definiálása (pl. mit kell tudni az egyes *unitok* feldolgozásának megkezdésekor, a *szakértő* mikor lép tovább, milyen mélységig tárgyal egy témát, mennyit hagyhat ki stb.)
– a tanulónak felkínálható eljárások leírása, azok hatáskörének, belépési pontjának kijelölése, az inputigények magyarázata, irányítása, bejárási, illetve elérési utak szekvenciájának ismertetése
– az egyes működések automatikus demonstrálása (tutor és help module megtervezése)
– a példafeladatok, laborszimulációk megoldásának követhető bemutatói.

2) Jelöljük ki az elérendő célt, ahol a globális tudásbázis szegmentálása során részfeladatként, a felhasználó pillanatnyi állapotához igazítva legyen kitűzve az
– eljárások orientációja
– az új irányok meghatározása
– a sikertelen kísérletek oknyomozása.

3) Adjuk meg
– az eljárások követhető magyarázatait
– az egyenértékű megoldások utalásait
– az alternatív vagy inverz módszerek tesztjeit
– esetenként készítsünk táblázatokat.

4) Készüljünk fel a hibákra, tárjuk fel
– a műveletek lehetséges hibáit
– a tévesztések magyarázatait és következményeit
– a kapcsolódó tévutak hamis irányait.

5) Tervezzünk folyamatos kapcsolattartást, ahol minden szinten meg kell oldani
– a visszacsatolást
– a tanuló modelljének folytonos korrekcióját
– a hibás inputok elemzésén alapuló kérdések generálását
– az önkontroll, a javítás lehetőségének módját
– a saját jegyzetelést, illetve annak lehívhatóságát.

6) Ellenőrizzük előfuttatással
– feltételeink bevezetésének jogosságát
– eljárásaink kapcsolatait
– értékelő rutinjaink egzaktságát
– a korrekciós utak helyességét.

Ez a felsorolás csupán egy a lehetséges felépítések közül. Minden ITS fejlesztő saját tapasztalataira hagyatkozva tehet hozzá, vehet el belőle az igényekhez és körülményekhez igazodva. Meggyőződésem: ahány rendszer, annyi módszer – miként azt az alábbiakban is láthatjuk.

Modelltípusok

Valamennyi számítógépes interakció alapvető problémája a felhasználót karakterizáló ismeretek teljes hiánya. Első lépésben tehát programunknak fel kell térképeznie a vele kommunikációs kapcsolatba lépő felhasználó személyiségét, tudásszintjét, problémamegoldó készségét. *Douglas M. Towne* és munkatársa, *Allen Munro* 3 modell-típust tart megvalósíthatónak.

1) Egy előteszttel felmérjük az adott problémakörhöz kapcsolódó tudásanyagot, ennek alapján – szigorúan a megtanítandó tananyag elsajátítási szintjeire vonatkozó – modellt generálunk a felhasználóról, s ehhez igazítjuk a monitorra kerülő részleteket.

2) Általános felméréssel tájékozódunk a felhasználó képességeiről, s erre alapozva szervezzük a kérdés-generálót, illetve a help-funkciókat, és ezután a szakértői modul a teljes ismeretanyagot feldolgoztatja.

3) Az előbbi két pont egyesített változata.

Wescurt és munkatársai már 1977-ben, a kognitív pszichológia kutatási eredményére támaszkodva kifejlesztettek egy, a tanulói tudásbázist közelítő analitikus programot, amelyben a rendszer a kapott inputokból folyamatosan módosította a student module-t, előrejelzést adott a várható haladási ütemről, az elsajátítás valószínű mértékéről és a továbblépési lehetőségekről. Egy másik közelítés a hibák elemzésére épül. Ezzel a témával a szakirodalom régóta foglalkozik. Például *Buswell* már 1926-ban hosszú listát állított össze a matematikai számításokban előforduló hibákról. *Burton* és *Brown* ugyancsak a matematika területéről készített „hiba-könyvtárat"(bug library) 104 darab típushibával, melyet a BUGGY, DEBUGGY, IDEBUGGY programokban használtak fel. A fejlesztés során kidolgoztak egy rendszert, amely generálni tudja a tanuló által vétett hibák struktúráját, s ezzel képes kijavítani azokat, illetve feladatokat kreálva a helyes szabály elsajátítását teszi lehetővé.

A student module egy újabb közelítése a hierarchikus tudásreprezentáció kategorizáló tulajdonságára épít. Itt az inputok elemzésekor az egyes részcélok elérését figyelik és a kérdés-generálás az aktuális szint elsajátítási mértékéhez igazodik. *Langley* és *Ohlsson* dolgozta ki a módszert, amelyet ACM *(Automated Cognitive Modeling)* diagnózisrendszernek neveznek. A modell továbbfejlesztett változata a DPF *(Diagnostic Path Finder),* amelyben nemcsak az elérendő célszinthez tartozó adathalmaz aktív, hanem az inputból felépíthető bejárási utakhoz (path-hoz) rendelhető tudáselemek is az elemző pufferbe kerülnek, amellyel már a várható haladási ütem, elsajátítási és alkalmazási szint is diagnosztizálható.

A kontextustól független természetes nyelvi elemzők (CFG parsher) döntési fa-struktúrájával rokon módszer a stratégiaelemző CIRRUS rendszer. Az input elemzése itt a ciklusmentes irányított gráfok éleihez rendelt (olykor komplex) adatelemekből épül fel, amelyben a fákat a szokásos „gyökér+lista" alakban ábrázolják. Minden beérkező adat módosít(hat)ja a fa szerkezetét, és a program képes megállapítani a felhasználó tervszerű vagy heurisztikus feldolgozási stratégiáját is. (Itt a heurisztikát, mint AI alkalmazási metodikát említjük, azaz nem a találgatásos kereső módszerre utaltunk.)

A student module-ok egyik elterjedt rendszere a lépésenként haladó. Ebben a metodikában minden lépésről külön értékelés készül, függetlenül az egységek feldolgozásakor végzett munka mennyiségétől és minőségétől, a továbbhaladásra jogosító tudásszint elérése egyenlő a teljes értékű, azaz a kiváló tudással. Az alkalmazás egyik kiemelkedő példája a WUSOR, melyben az elért szintről az inputokból épített genetikus gráf elemzésével kapunk képet. A WUSOR egy játék, amelyben labirintusok sokasága építhető fel az inputok alapján, mivel az út minden fordulójánál az új irány megválasztása egy-egy kérdés megválaszolásának eredményétől függ. A rendszer legnagyobb előnye, hogy az előrelépés nem egy deklaratív, lineáris struktúrájú forgatókönyv előírásai alapján történik, hanem az épülő gráf aktuális éleihez rendelt információelemek ismertségi fokától függ.

Technikai jellegű kísérlet a felhasználó tudásszintjének nyomon követésére a DIAG program (*Boohan*, 1992). A modell Shipstone ötletét alkalmazza, amely egy áramköri kapcsolásként reprezentálja az aktuális tudás szintjét. Ha egy áramkörbe párhuzamosan kötünk két izzót – az egyik az elsajátítandó tudást, a másik a felhasználó tudását, az áram pedig az információ áramlását jelképezi –, akkor különböző ellenállások be-, illetve kiiktatásával elérhetjük, hogy a két izzó egyforma intenzitással égjen, azaz a két szint – a szakértői és az elsajátítási – egyező legyen. A söntök a részinformációk és a szükséges adatok halmazát jelképezik, tehát a mellékkörökben az azonos áramerősséget a söntök kiiktatásával lehet elérni. A modellhez tartozó egyszerű számítási művelet általános alkalmazást tesz lehetővé.

Kurt Van Lehn húsz kidolgozott modelltípusról tesz említést, és még legalább ennyi fejlesztésén dolgoznak. A különböző algoritmussal és memóriamodellel dolgozó metodikák abban egyeznek meg, hogy mindegyik a felhasználótól származó inputból épít fel egy szabályozó keretet, amellyel valamilyen rendszerbe szervezhető az információ.

Hogy ezek közül a módszerek közül melyik reprezentálja jobban az emberi gondolkodás szerkezetét, illetve az adathalmazok megfelelő szerinti rendezésének elvét, azt a kitűzött cél elérésének sikeressége határozza meg.

Közelítések AI-technikákkal

Az AI *(Artifical Intelligence)* mint önálló tudományág, a humán információfeldolgozási stratégiák gépi szimulálásának lehetőségeit kutatja. Pontos körülírását *Dan W. Patterson* adta meg 1990-ben. „Az AI az informatika *(számítógép tudomány)* egyik ága, amely tanulmányozza a gépi intelligencia lehetőségeit. Intelligens tanító/tanuló rendszereket, feladatgenerálókat fejleszt, kutatja azokat az algoritmusokat, amelyekkel a számítógép 'megtanítható' intelligens gondolkodásra, következtetések levonására, a bennünket körülvevő világ érzékelésére, a természetes nyelvi és vizuális információk feldolgozására és a válaszok humán elvárások szerinti generálására." Az AI felosztását az alábbi nyolc fő kutatási területben jelölhetjük meg:

– természetes nyelvfeldolgozás (nyelvelemzők – Tomita Parsher)
– tanulás (ITS – Intelligent Tutoring Systems)
– automatikus programozás (C.A.S.E. tools – object oriented programming)
– tervezés (CAD rendszerek)
– beszédfelismerés (Ensemble Interval Histogram Model – Ghitza modellje)
– észlelés-érzékelés (digitális képfelismerők)
– megértés (metanyelvek, szemantikus/pragmatikus elemzők)
– ideghálózati modellezés (connectionism, neural network models).

A FORMERS interaktiv „zeneszerző" program *(Xavier Rodet és Pierre Cointé)*, a Starplan képfelismerő tervező program *(Ron Siemens és Jay Ferguson Ford)*, a PRIDE ny.á.k. tervező program (Xerox Park), az EURIKSO ITS heurisztikára épülő tanító program kiváló példái az AI technikák eltérő implementációban történő alkalmazására. Oktató programok, értékelő rendszerek fejlesztéséhez elsősorban a felismerő, azonosító és osztályozó algoritmusokat alkalmazhatjuk. Annak eldöntésére, hogy az AI kutatók által kitűzött célok reálisak-e vagy utópiák, nem vállalkozhatunk. Azonban – függetlenül az elhangzó pro és kontra érvektől – valljuk, hogy a működő modellek, algoritmusok, a kidolgozott specifikus AI számítási technikák nélkülözhetetlenek az ITS területén dolgozók számára. A nagyon kevés, magyar nyelven is hozzáférhető irodalomból *Fekete*, *Gregorics*, *Nagy* könyve ad áttekintő ismertetést erről a területről.

Néhány számítási módszer különösen érdekes, ezért alábbiakban egy tetszőleges feladat állapotterén értelmezhető műveletek formuláit tárgyaljuk (állapottér = az adatbázis összes lehetséges kapcsolatainak permutációjával megvalósítható esethalmaza).

Vegyünk például egy tudásbázist, melyben az egyes tudásszinteket szegmentált elemekben tároljuk. A tanuló modelljének felépítéséhez a beérkező inputok (az elsajátítás mértéke) és a súlyozott készlet (a megtanítani kívánt anyag) közötti korrelációt számítjuk ki, amit súlyozott összegnek, szintmutatónak értelmezhetjük és a

$$S_j = \Sigma w_{ij} x_i \quad (i=1\text{-}n)$$

formulával számítjuk, ahol S_j az input j tagjára súlyozott összeg, w_{ij} az i- és j-edik tag közötti kapcsolat súlya, x_i az i-edik elem értéke.

Kohnen ezt a súlyozott összeget két vektor, **x** és **y** egymás közötti, Euklideszi távolságában határozza meg az alábbi módon:

$$[\Sigma(x_i - y_i)^2]^{1/2}$$

melyet *Minkowski* általánosított:

$$(\Sigma(x_i - y_i)^n)^{1/n}.$$

Ebben a formulában, ha n=2 (amit sok szerző Manhattan mértéknek nevez), Kohnen képletét kapjuk. A relatív orientáció számításakor azonban nem tudjuk kezelni azokat az eseteket, amikor a két vektor párhuzamos, tehát $\Theta=1$, vagy ortogonális, azaz $\Theta=0$, ame-

lyet két valós és igaz állítás logikailag nem kapcsolható, illetve két valós és igaz állítás logikailag ellentmondó eseteként értelmezhetünk.

Ha a fenti diszkrét értékű vektorokat és komponenseiket bináris reprezentációban kívánjuk megadni, használjuk a Hamming-távolság formalizmusát, ahol a két vektor távolságát úgy kapjuk meg, hogy egyszerűen az exluzív or (XOR) műveletet futtatjuk végig minden komponensen:

$$\delta = \Sigma xor(x_i,y_i).$$

Ha az agyi ideghálózat modellezését tűzzük ki elérendő célként, el kell döntenünk, hogy diszkrét vagy folytonos közelítéssel ábrázoljuk az egyes folyamatokat. Sokan úgy gondolják, hogy az egyes információelemek közötti összefüggések leírásához használt mutatók túlságosan sokszor eredményeznek korrigálhatatlan kerekítési hibákat. Ennek kiküszöbölésére állapodjunk meg abban, hogy az input aktivációs jelsorozatát folytonosnak tekintjük. Ekkor, a fuzzy logika módszereivel jutunk eredményre, ahol a klózhalmaz minden eleme valamilyen intervallumon folytonos függvénnyel írható le. Tekintsük a bejövő információhalmaz szignálját, az S(x(t)) függvényt mint a jel időfüggését, és fogadjuk el, hogy ezt az

$$S(x) = \frac{1}{(1 + e^{-cx})}$$

függvény írja le, ahol a $c > 0$ és a függvény monotonitása biztosítja a mindig pozitív eredményt. Ekkor az input aktiválta i-edik információelem valószínűsíthető helyét az általánosított fuzzy-Gauss kalkulussal kereshetjük meg:

$$S_i(x) = -\left[\frac{1}{(2\sigma_i^2)\,\Sigma(x_j - \mu_{ij})^2}\right]$$

ahol $\mathbf{x} = (x_1,...,x_n) \in R^n$ az input aktivációs vektora, σ_i^2 a szórásnégyzete, $\mu_i = (\mu_{i1}, ..., \mu_{in})$ pedig az irányvektor. *L.A. Zadeh* formuláját mint egy hálóbeli elem megtalálásához alkalmazható számítást kell még figyelembe vennünk:

$$e(x,y)=max(min(x,y),min(1-x,1-y))$$

ahol x és y a 0 és 1 közötti folytonos intervallumon reprezentálja egy-egy klóz igazságértékének mértékét. Ennek bemutatására kövessük végig például a modus ponens fuzzy reprezentációját. Legyen ***A***, ***A1***, ***B***, ***B1*** egy-egy kijelentés (állítás) a fent említett fuzzy reprezentációban meghatározva.

Általánosan:

premissza: $x = \boldsymbol{A1}$
implikáció: ha $x = \boldsymbol{A}$ akkor $y = \boldsymbol{B}$
konklúzió: $y = \boldsymbol{B1}$.

Kibontva:

premissza: ez az izzó nagyon világít
implikáció: ha egy izzó világít,
akkor az áramot fogyaszt
konklúzió: ez az izzó
nagyon sokat fogyaszt.

Amennyiben Zadeh következtetési szabályát szeretnénk alkalmazni, definiálnunk kell a relációkat. Képezzük *A* és *B* Descartes-szorzatát, ***A X B***-t (kiolvasva: a kereszt bé), amely megadja az adott halmazokból létrehozható összes (a,b) párt, ahol $a \in A$ és $b \in B$. Nyilvánvaló, hogy *A* és *B* halmazok részei az ***A X B***-nek. Definiáljuk továbbá a bináris fuzzy relációt, **R**-t, tehát a Descartes-szorzat eredményét mint implikációt, a tartalmazási függvény két paraméterével meghatározhatóan $e_R(a,b)$. Legyen például $A=B=R$ a valós számok halmaza és legyen **R** := „nagyobb mint" reláció. Ekkor a tartalmazás függvényét az alábbiak adják:

$$e_R(a,b) = \begin{cases} 0 & \text{ha } a \le b \\ (1+(a-b)^{-2})^{-1} & \text{ha } a > b \end{cases}$$

Most legyen X és Y két univerzum, bennük A, illetve B fuzzy halmaz, X -ben és X **X** Y -ban. Ezekkel a fuzzy reláció: $\mathbf{R}_A(x)$, $\mathbf{R}_B(x,y)$, és $\mathbf{R}_C(y)$ X-ben, X**X** Y-ban és Y-ban. Ezek után a következtetési szabály alapján az alábbi egyenlethez jutunk:

$$\mathbf{R}_C(y) = \mathbf{R}_A(x) \mathbf{o} \mathbf{R}_B(x,y) = \max_x \min\{e_A(x), e_B(x,y)\}$$

ahol **o** jelöli A és B kompozícióját. Behelyettesítve a teljes példa:

$X = Y = \{1,2,3,4\}$

A = {kisebb} = (1/1),(2/0.6),(3/0.5),(4/0.2)

R = körülbelül egyenlő

y		1	2	3	4
x	1	1	0,5	0	0
	2	0,5	1	0,5	0
	3	0	0,5	1	0.5
	4	0	0	0,5	1

Alkalmazva a max-min rendezési szabályt:

$\mathbf{R}_C(y) = \max_x \min\{e_A(x), e_B(x,y)\} =$
$\max_x$ {min[(1,1),(0.6,0.5),(0.2,0),(0,0)],
min[(1,0.5),(0.6,1),(0.2,0.5),(0,0)],
min[(1,0),(0.6,0.5),(0.2,1),(0,0.5)],
min[(1,0),(0.6,0),(0.2,0.5),(0,1)]} =
$= \max_x$ {min[1,0.5,0,0],[0.5,0.6,0.2,0],[0,0.5,0.2,0],[0,0,0.2,0]} =
{[1],[0.6],[0.5],[0.2]}.

Tehát az eredmény:

$\mathbf{R}_C(y)$ = {(1/1),(2/0.6),(3/0.5),(4/0.2)}.

A levezetés példa a modus ponens fuzzy-beli reprezentációjára:

premissza: x kicsi
implikáció: x és y körülbelül egyenlő
konklúzió: y több vagy kevesebb mint kicsi.

A bemutatott eljárások, számítási formulák önmagukban nem túl sokat mondanak. Azonban valamennyi közelítés teljes kifejtése, példával történő illusztrációja meghaladja e dolgozat kereteit, így csak javasolni tudom a hivatkozott irodalmat, ahol mindegyik megtalálható az összes vonzatokkal együtt. Alapos megfontolások után a fenti leírásokkal szemben az egyetlen, ám súlyos probléma – a modellezhetőség szempontjait tekintve – a következő: Ha a rendszer általánosítható, vagy legalább néhány bonyolultabb feladat kimunkálására alkalmas, akkor a megoldás keresése során kombinatórikus robbanás következhet be a létrehozható kapcsolatok logaritmikus növekedése miatt. Ha pedig az optimális megoldás helyett kénytelenek vagyunk heurisztikát alkalmazni, és így megelégedni a kielégítő megoldással, esetenként vállalva, hogy meg sem találjuk a kivezető gráfot, elveszítjük a célt, a sokszempontú, általánosítható modell kialakításának lehetőségét. *Matthew Zeidenberg* fenntartása tovább bonyolítja a problémát, ugyanis szerinte, ha az objektumokat összes attributumaikkal ábrázoljuk, a létrehozható kapcsolatok közül a gép – alkalmas algoritmus híján – nem tudja kiválasztani a valóságban nem előforduló eseteket. Ha megkíséreljük egyetlen esetre redukálni a kapcsolatrendszert, megoldhatatlan problémába ütközünk. Például a DUCS rendszer szelektorának memóriaigénye

$$4F^*2^R$$

ahol F = a lehetséges attributumkapcsolatok száma (Zeidenbergnél: microfeatures's connect) R = az objektumok lehetséges kapcsolatainak száma.

Könnyen belátható, hogy a formulából számolható memóriaigény viszonylag alacsony objektum- és attributumszám esetén is aránytalanul nagy, sőt megvalósíthatatlan méreteket ölt.

Nem kell elvetni mégsem ezeket a közelítéseket, mert speciális esetekre, jól szegmentálható tudáselemekkel (ha az nem túl sok), nagy pontossággal alkalmazhatók. Ha a kidolgozott formalizmushoz készítünk egy keretrendszert, amely kezeli a kapcsolatszinteket, illetve vezeti a szükséges nyilvántartásokat, esetenként célravezető eszközökké válhatnak.

Több szempontból is hasznos lenne, ha a keret és a bázis felépítésekor egyező logikájú struktúrában gondolkozhatnánk. Melyik az a programozási nyelv, amelynek szemantikája lehetővé teszi a humán gondolkodás asszociációs képességének szimulálását?

A LISP *(LISt Processing language)* mint funkcionális programozó nyelv, listafeldolgozó, szimbolikus kifejezésekkel végzett műveletekre alkalmas nyelv. Az 1950-es évek végén dolgozták ki a Massachusetts Institute of Technology munkatársai, a Man and Computer program keretében. A LISP nem szabványosított nyelv, de az egyes változatok alapelvei közös bázison nyugszanak. Három nagy család köré csoportosítható a forgalmazott verziók mindegyike: INTERlisp, COMMONlisp, MAClisp. A nyelv lényege, hogy a szimbolikus és numerikus „atomok" (a nyelv legkisebb elemei) tetszőleges sorozatú listái különböző mélységű gráfokkal ábrázolva, állapotreprezentációk. Ezen az „esetfelületen" különböző függvényeket értelmezhetünk, azaz a listákat lebonthatjuk, újraépíthetjük, rendezhetjük a kívánalmaknak megfelelően. Mint említettük, a nyelv nincs standardizálva, így alkalmazása helyett egy másik, hasonló elvekre alapozó programozói nyelvet javasolunk.

A PROLOG

Ez a nyelv az ITS programozók csaknem kizárólagos nyelvévé vált. Az elsőrendű predikátumkalkulus részhalmazaként nyilvántartott definitklóz nyelvtan, a logikai programozás máig leghatékonyabb eszköze. A PROLOG-ot, mint egy, a metamorfózis nyelvtanokban használt Q-rendszer továbbfejlesztett változatát, egy procedurális értelmezésű DCG (*definit clause grammar*) rendszerként használjuk, amelynek definícióhalmazát az alábbi köznyelvi tömörítéssel jellemezném:

„Helyettesítsd a gráfok így és így definiált halmazát egy másik, úgy és úgy definiált halmazával!"

A PROLOG alkalmas lehet a magyar nyelvű természetes nyelvi interface kialakítására, amely a tanuló modelljének megalkotásához elengedhetetlen. Tudjuk, hogy a PROLOG-ban a ragnyelvtanokkal rokonságot mutathatunk ki, mert a CFG-beli nemterminálisokat argumentumokkal bővíthetjük, ahogy nyelvünkben toldalékolunk, sőt, mint CFG nyelvleíró eszköz, a magyar nyelv kötetlen szórendjét is elemezhetjük (CFG=*context free grammar*).

A PROLOG ITS-beli alkalmazásának egyik legkorábbi példája a WEST, amely egy matematikai alapműveleteket tanító rendszer. A program egy egyszerű, de izgalmassá tehető játékon alapul, melyben a tanuló egy „fogathajtó" versenyen, vagy ha úgy tetszik autóversenyen méri össze számolási készségét a computerrel. Minden „futam" n egységnyi út bejárását jelenti (n implementációtól függő érték), ahol a gép generálta véletlenszámokkal végezhető aritmetikai műveletekkel kell megszabni az előrejutás optimális ütemét. A játékszabályok megengedik a kiszorítást, a pályán sok rövidítés és akadályvan, vagyis komoly tervezést, stratégiát követelő rendszer. A deklaratív tudásbázist imitáló program tutorja, ez az egyszerű, természetes nyelvi interface a PROLOG alkalmazásának lehetőségeit mutatja be.

Ahogy minden ember gondolatvilága egyedi, úgy minden program struktúrája is egyedi. Egy-egy feladat megoldásához több úton is eljuthatunk, ahogy minden megoldható programozási feladathoz is különböző Turing-gépek készíthetők, ezért nem kötelező választás egyetlen nyelv sem. Lehetnek speciális közelítések, speciális programozó nyelvvel, sőt, készülhet csak a feladatra koncentrált metanyelv is.

A felhasználóval fenntartandó párbeszédes kapcsolat kidolgozásakor az input kiértékelése egyben minősítés is, hiszen a válasz(ok) illeszthetősége, strukturáltsága, adekvátsága valamilyen nyelvi keretből építkezik.

Interface fogalomhálón alapuló nyelvtannal

Legyen egy tetszőleges természetes nyelv. Ehhez járuljon egy egyértelmű leíró jelsorozat, amellyel denotálhatjuk a nyelv összes morfémáját. Szükségünk van egy CF *(Context Free)* nyelvtanra, illetve ennek egy attributumokkal kiegészített változatára, az ún. AG-re (*Affix Grammar* – attributum nyelvtan). Az elképzelt rendszerben az attributumok egymással kapcsolatban lehetnek, függhetnek egymástól, eredhetnek egymásból. A metanyelvekben ezek a függések definitek, kidolgozott, egyszerű algoritmusokkal leírhatók. Azonban a természetes nyelvekben egy objektumról leírható jelöletek nem konkretizálhatók annyira egyértelműen, mint a metanyelvekben. Nem dolgozhatunk definit attributum-halmazrelációkkal, illetve egyértelmű konjunkciókkal. Az attributumok közötti lehetséges kapcsolatok rendszere csak tanulással építhető ki, illetve a bejárási utak közül a kontextushoz adekvát válasz megtalálása csak korábbi esetek valamilyen szinten való megélése, azonosítása alapján történik.

Ez a folyamat (a tanulás, ill. tapasztalás), egy folytonosan változó, modellezhető kapcsolatrendszer menedzselése. A modell megkísérli szimulálni az emberi gondolkodást, illetve ennek transzlátumát a nyelvre vetítve. Ezt fogalomháló-építő, -törlő, -kapcsolatteremtő, -objektumalkotó operátorok működtetésével oldja meg. A feladat formalizmusát, illetve az algoritmusok alkalmazhatóságát olyan matematikai-algebrai tételek támogatják, mint a halmazelméleti relációk generálta lezárások (Galois kapcsolatok) kísérő struktúrái, amelyek jelen interpretációban a *fogalomhálók*.

Kettős elemzésre van szükségünk:

1. szóbázisú elemző, amely az affix nyelvtanoknak egy új megfogalmazása lehet; szótövek és végződések rendszereinek belső elemzése.
2. mondatbázisú elemző, amelyben a szóbázisú elemzés során inicializált mikrostruktúra-kapcsolatok kiértékelése történik.

Ezeken felül létezhet egy ún. blokkelemző (szövegelemző) alrendszer, amely az interszentenciális kapcsolatok helyét, módját, milyenségét határozza meg. Ez az alrendszer a stilisztikai következtetéseken túl szemantikai értékeléseket is ad.

A rendszer az elemzés mellett, megfelelő kiépítéssel, természetes nyelvű szövegek generálására is alkalmassá tehető. A rendszer mint humán szövegelemző-szerkesztő szimuláció, egy program inputjainak értékelésére, illetve outputjainak szervezésére alkalmazható. Ha a tárgyi szakértő struktúrája ezen az alapon épül fel, a „tanulás" folyamata azonos lesz a működés során rögzíthető struktúrák megjelenési formájával, következésképpen a megtanítás hatékonysága a lineáris predikció módszerével megjósolható. A teljes kiépítés célspecifikációit minden fejlesztő maga határozza meg, a műveleti struktúrát pedig az alábbiakban mutatjuk be.

A javasolt modell

Legyen egy rendszer adatbázisában megtalálható törvény a tömegmegmaradás törvénye:

Zárt rendszerben végbemenő reakció során a kiindulási anyagok együttes tömege megegyezik a reakcióban keletkező anyagok együttes tömegével.

A tömegmegmaradás törvénye zárt fogalomháló, amelynek önálló modulként kell állnia a tudásreprezentációban, ahol a modul egyes csomópontjai (a nodok) relációkkal kapcsolódnak egymáshoz.

A törvény mint megtanítandó egység szerepel rendszerünkben, abból az alapállásból kiindulva, hogy a felhasználó magát a definíciót nem, de önálló elemeit, nodjait már korábbról ismeri. (Ellenkező esetben természetesen visszautalható a feladat a fogalomismertető rutinhoz!) A bázisban szereplő fogalomháló relációi, kapcsolatrendszere magában hordozzák a törvény ismeretének kritériumait, azaz, ha feltárjuk a belső struktúrát, pontosan meghatározhatjuk az elvárások objektív kívánalmait. Bontsuk fel részhalmazokra a törvényt, amelyekből generálhatjuk a nodokat, majd definiáljuk a relációkat:

Legyenek a részhalmazok típusai: adatok, tények, fogalmak, relációk.

– adatok: kiindulási anyagtömeg, keletkezett anyagtömeg {A}, {B}
– tények: reakció {C}
– fogalmak: zárt, rendszer, reakciótípusok {D},{E}, {F}
– relációk: egyenlő, szumma, konjunkció {=}, {Σ}, {∩}

Gépi feldolgozáshoz, a nyelvi elemek jelenlétének, hiányának, szükségességének eldöntéséhez természetes nyelvi elemzőt használunk.

Alkalmazva a nodok rövidítéseit, a törvény logikai leírása:

premissza: $C(a_1, a_2,...a_n)$ $a_n \in A$
implikáció: ha $D(C) \in E(F)$, akkor $A = B$
konklúzió: $C(a_1, a_2,...,a_n) = B(b_1,b_2,...,b_n)$

Kibontva:

premissza: $a_1, a_2,...a_n$ anyag reakcióba lép
implikáció: ha a rendszer zárt,
akkor a belső változások eredője 0
konklúzió: a C reakcióban $\Sigma a_n = (\Sigma b_n)$
anyag keletkezik

A törvényt reprezentáló háló-modul generálásához képezzük az egyes nodok és azok attributumainak halmazpárjait a Descartes-szorz'ssal, majd végezzük el a részbenrendezések és lezárások műveleteit minden halmazpáron, ami után a generált nodok:

$N0 = <\{a_1, a_2, a_n, b_1, b_2, b_n, C, D, E\}, \{\}>$
$N_1 = <\{a_1,a_2,a_n\}, \{\Sigma\}>$
$N_2 = <\{b_1,b_2,b_n\}, \{\Sigma\}>$
$N_3 = <\{a_1,a_2,a_n,b_1,b_2,b_n\}, \{C\}>$
$N_4 = <\{\}, \{C, D, E\}>$

Ebből építsük fel a hálót

A kész hálóból nemcsak maga a törvény olvasható ki, hanem a részhálók tartalma is. Például az N_0 N_1, N_2, N_4 nodok generálta részháló a lehetséges reakciók típusait mutatja meg, az $a_1,a_2,...,a_n$ és a $b_1,b_2,...,b_n$ anyagok ismeretében N_1, N_2, N_3 részháló a reakció lejátszódásáról nyújt információt.

A tanuló tényleges tudásáról akkor ítélhetünk objektíven, ha a pontosan definiált tudáselemek (nodok) ismertségét rögzíteni tudjuk. Egész munkánk során tehát legfontosabb momentum a bevezetett témakör felbontása. Meg kell adni minden olyan elemet, amely az elsajátítandó részhez rendelhető, ismertetni kell azokat a kalkulusokat, amelyek megadják az elemek közötti leképezéseket. Az értékelés során csupán az ismert elemek és azok egymáshoz rendelésének szabályait, pontosabban azok meglétét vagy hiányát elemezzük. Amennyiben a szükséges és elégséges node-relációval felépíthető a megtanítani kívánt háló, a tudás elégségesnek mondható. A besorolási kategóriák természetesen feladatfüggőek, s kidolgozásukhoz a pedagometria számtalan lehetőséget kínál.

IRODALOM

Boohan, R.: DIAG: a Program to Diagnose Students' Conceptual Models in Science = Journal of Computer Assisted Learning, 8. 1992. 206-220. p.

Brown, J.S. – Burton, R.B.: Diagnostic Models for Procedural Bugs in Basic Mathematical Skills = Cognitive Science, 2. 1978. 155-192. p.

Brown, J.S. – Van Lehn, K.: Repair Theory: A Generative Theory of Bugs in Procedural Skills = Cognitive Science, 4. 1980. 155-192. p.

Burton, R.B.: DEBUGGY: Diagnosis of errors in Basic Mathematical Skills = In *D.H. Sleeman – J.S. Brown* (Eds.) Intelligent Tutoring Systems, Academic Press, New York. 227-282. p.

Burton, R.B. – Brown, J.S.: An Investigation of Computer Coaching for Informal Learning Activities. In *D. Sleeman – J.S. Brown* (Eds.): Intelligent Tutoring Systems. 1982. Academic Press, New York, 79-98. p.

Buswell, G.T.: Diagnostic studies in arithmetic. IL: University of Chicago Press. Chicago, 1926.

Colmerauer, A.: Metamorphosis Grammars. In L. Bolc (Ed.) Natural Language Communications with Computer. Springer Verlag, München. 1978.

Fekete, I., – Gregorics, T. – Nagy, S.: Bevezetés a mesterséges intelligenciába. LSI Oktatóközpont, Budapest. 1990.

Fletcher, J.D.: Modelling of Learner in Computer-Based Instruction = Journal of Computer-based Instruction (1.) 1975. 118-126. p.
Kohnen, T.: Self-Organization and Associative Memory. Springer Verlag. Berlin. 1988.
Goldstein, I.: The Genetic Graph: A Representation for the Evolution of procedural Knowledge. In: *D. Sleeman – J.S. Brown* (Eds.), Intelligent Tutoring Systems 51-77. p., Academic Press, New York. 1982.
Kosko, B.: Neural Networks and Fuzzy Systems. Prentrice-Hall International Editions. 1992.
Koster, C.H.A.: Affix Grammars. In J.E.L. (ED.) Algol 68 Implementation. North-Holland, Amsterdam. 1971.
Langley, P. – Ohlsson, S.: Automated Cognitive Modelling Proceedings of American Association of Artifical Intelligence 193-197. p. Los Altos, CA: Morgan Kaufman. 1984.
Ohlsson, S. – Langley, P.: Identifying Solution Paths in Cognitive Diagnosis. (Techn. Rep. CMU-RI-TR-84-7.) Pittsburgh,PA:Carnegie Mellon University,Robotic Institute. 1985.
Lehnert, W.G.: A Conceptual Theory of Question Answering. Lawrence Erlbaum Associated Publisher, Hillsdale. 1977.
Patterson, D.W.: Introduction to Artificial Intelligence and Expert Systems. Prentice-Hall International Editions. Engelwood Cliffs, New Jersey. 1990.
Squire, R.L.: Memory and the Hippocampus: A Synthesis From Findings With Rats, Monkeys and Humans. Psychological Review vol. 99. 1992. num. 2. 195-231. p.
Towne, D.M. – Munro, A.: The Intelligent Maintanance Training System. In> Intelligent Tutoring System (Eds.) *J. Psotka, L.D. Massey, S.A. Mutter.* Lawrence Erlbaum Ass. Pub. (18.) 1988. 479-530. p.
Van Lehn, K.: Bugs are not Enough: Empirical Studies of Bugs, Impasses and Repairs in Procedural Skills = The Journal of Mathematical Behavior, 3(2), (1982). 3-71. p.
Van Lehn, K.: Felicity Conditions for Human Skills Acquisition: Validating on AI-Based Theory (Tech. Rep. CIS-21) Palo Alto, CA: Xerox Palo Alto Research Center. 1983.
Van Lehn, K. – Garlick, S.: Cirrus:An Automated Protocol Analysis Tool. In P.Langley (Ed.), Proceedings of Fourth Machine Learning Workshop. Los Altos, CA: Morgan Kaufman. 1987.
Van Lehn, K.: Student Modeling. In Foundations of Intelligent Tutoring Systems, Lawrence Erlbaum Associates Publishers, Hillsdale. 1988. 55-78. p.
Wescurt, K.T. – Beard, M. – Gould, L.: Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. Proceedings of the Association for Computing Machinery. 1977. 234-340. p.
Wille, R.: Restructing Lattice Theory:an Approach Based on Hierarchies of Concepts. In: *I.Rival* (Ed.): Ordered sets. Reidel, Dordiecht, Osten 1982. 445-470. p.
Wille, R.: Tensorial Decomposition of Concept Lattices. In: Technical Report 823. 1984. Technische Hochschule, Darmstadt.
Wille, R.: Bedeutungen von Begriffsvebanden. In: *B.Ganter – R.Wille – K.E. Wolff:* Beitrage zur Begriffsanalyse. B.I.-Wissenschattsverlag. 1987.
Zadeh, L.A.: Outline of a New Approach to the Analysis of Complete System and Decision Processes = IEEE Transactions on System, Man and Cybernetics SMC-3, 1973. (January) 22-24. p.
Zeidenberg, M.: Neural Networks in Artificial Intelligence. Ellis Horwood 4. 1990. 138-156. p.

Helyreigazítás

Az Iskolakultúra 1993/19. számában megjelent Térbeli alakzatok ábrázolása számítógéppel című cikkbe becsúszott néhány hiba.

A 8. oldal második bekezdésének utolsó mondata helyesen:

Ezzel lényegében megadjuk azokat a transzformációs képleteket, amelyekkel a tér egy P(x,y,z) pontjához hozzárendeljük ennek a képernyőre eső P'(a,b) vetületét.

A 15. oldal 2. és 3. sora helyesen:

```
if W <: '&' then rajzol else bip;
if d> 30 then begin persp:=false; d:=10; end
```

Sajnálatos módon lemaradt a programlista utolsó néhány sora, a főprogram is:

```
begin
  fejlec;
  grafika;
  init;
  fut;
  closegraph;
  iras (12,30,14,'Viszontlátásra')
  while not keypressed do'
end.
```