CAS automata elméleti felhasználásának didaktikai kérdései

Maróti György

1 Előzmények

A Maple nyitott architektúrájának jellemző vonása, hogy eljárásai a C nyelvű mag (kernel) kivételével a Maple saját nyelvén készültek, és nagy részük a felhasználók által fejlesztett csomagokban (package) található. A rendszerhez eleddig fejlesztett csomagok száma meghaladja a kilencvenet. Ezek számát kívánta növelni az automata elméleti modellezést támogató `aut` csomag, bevonva ezzel az automata elméletet a Maple-lel kezelhető területek körébe.

Mint minden Maple csomag, az `aut` is adatstruktúrák és azok létrehozását, valamint a velük való különböző manipulációkat támogató eljárások összetartozó együttese. Az aut számos új adattípust vezet be, többek között a `automaton` típust véges nemdeterminisztikus automatákra, a `deterministic` és `complete` típust determinisztikus illetve teljes automatákra, a `regular` típust reguláris kifejezésekre. A csomag mintegy 35 eljárása gyakorlatilag lefedi a klasszikus automata elmélet eredményeit a jól ismert automata konstrukcióktól, az automaták minimalizációján és a reguláris kifejezések egyszerűsítésen keresztül a Kleene tételig, az automaták analíziséig és szintéziséig. A 2500 feletti forrássorban teste öltő csomag 18 eljárása segíti az automaták létrehozását, a velük való manipulációt, valamint a különböző reprezentációkban megjelenő vizualizációt. A reguláris kifejezések kezelését 7 eljárás, míg az automata konstrukciókat további 8 eljárás támogatja.

2 Célkitűzések és megvalósítás

A jelen dolgozat célja megtalálni a választ két alapvető kérdésre.

1. *Használható-e, és ha igen, akkor milyen módon, az aut csomag automata elméleti kutatásokban?*
2. *Használható-e az aut csomag az automata elmélet oktatásának segédeszközeként, és ha igen, melyek azok a didektikai módszerek, amelyek mentén a haladva a csomag megkönnyítheti az automata elmélet fogalmainak, tételeinek tanítását és segítséget nyújthat az automata elméleti problémamegoldás elsajátításában?*

A kijelölt céloknak megfelelően a dolgozat hármas felépítést mutat. A dolgozat első, informatikai orientáltságú fejezete magát az `aut`

csomagot mutatja be. A második, elméleti részben néhány, a csomag felhasználásához közvetlenül köthető automata elméleti eredmény bizonyítunk. Végül a dolgozat egy matematika-didaktikai fejezettel zárul, ahol didaktikai módszereket dolgozunk ki az aut csomag oktatásban való hatékony felhasználására.

Az csomag részletes, kimerítő ismertetése terjedelmi okokból lehetetlen feladat, ezért első sorban annak használatára fókuszálunk, miközben bevezetjük a szükséges automata elméleti fogalmakat és a dolgozatban használatos jelöléseket. Nem törekedhetünk minden eljárás átfogó bemutatására, mint ahogy nem törekedhetünk a bemutatandó eljárások összes opciójának ismertetésére sem. Amit bemutatunk az nem más, mint a csomag eljárásainak tipikus használata.

Kitérünk azonban a különböző fogalmak Maple-beli implementációjára, ismertetjük a legfontosabb automata konstrukciókat (részhalmaz konstrukció, szorzat konstrukció stb.), majd megvizsgáljuk a csomag konstrukciós motorjának, a Compose eljárásnak az algoritmusát és elvégezzük az algoritmus komplexitás vizsgálatát.

A CAS kutatásokban betöltött szerepének hasznossága napjainkra már evidencia. Mind az általános célú, mind pedig célfeladatokra fejlesztett komputer algebrai rendszerek egyre szélesebb körben nyernek létjogosultságot a tudományos kutatás különböző területein. A dolgozatunk egyik célja éppen az, hogy az automata elméleti kutatásokat is felsorakoztassa ezen területek sorába. Ennek megfelelően a dolgozat második része néhány az **aut csomag használatához köthető automata elméleti eredményt** tárgyal.

Elsőként egy általános tételt bizonyítunk automata konstrukciók kompozícióira. Majd rátérünk automaták irányíthatóságának eldöntésére **Imreh és Steinby** által kidolgozott algoritmus implementálásának tárgyalására. Megmutatjuk, hogy az automata irányító táblájának felhasználásával az algoritmus alkalmassá tehető az irányíthatóság eldöntése mellett irányító szó meghatározására, sőt kommutatív automaták esetén minimális hosszúságú irányító szó előállítására is.

Az irányító tábla ismeretében lehetőség nyílik egy az irányítható automatákhoz társított automata bevezetésére, melyet μ-automatának nevezünk. Megmutatjuk, hogy **a μ-automata irányíthatósága az automata irányíthatóságának szükséges és elégséges feltétele**. A μ-automaták több bemenő jellel ám lényegesen rövidebb irányító szavakkal rendelkeznek. Nyitott kérdés, hogy a μ-automaták viszgálata milyen szerpet játszhat az automaták irányíthatóságának vizsgálatában.

A dolgozat utolsó része tárgyalja az aut csomag oktatásban való felhasználásának didaktikai kérdéseit. Részletesen elemezzük és Maple-lel illusztráljuk a reprezentációk oktatásban való használatára vonatkozó NCTM (National Council for Teaching Mathematics) 2000-ben közzétett ajánlását megfogalmazó **négyes szabályt** (Rule of Four). Rámutatunk a különböző reprezentációk többszörösségére, a reprezentációk átfogalmazhatóságának és átjárhatóságának fontosságára, a realisztikus reprezentáció kognitív hatékonyágot befolyásoló szerepére. Végül a **javaslatot teszünk egy a négyes szabálynál általánosabb didaktikai elv**

megfogalmazására, melyet a többszörös reprezentáció elvének nevezünk.

Az itt javasolt didaktikai elvek konkrét alkalmazásaként kidolgozunk két Maple munkalapot, egyet az automata elméleti fogalmak tanítására és egy másikat az **automataelméleti probléma-megoldás tanítására**. A már jól bevált didaktika elvek (aktivitás elve, irányított felfedezéses tanulás, stb.) értelemszerű alklamazása mellett javaslatot teszünk, és egyben példát is adunk a Maple munkalapok didaktikai struktúrájának felépítésére.

3 A disszertáció tézisei

3.1 Automata konstrukciók kompozíciója

3.1.1 Definíció

Tekintsük a $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ és a $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$ automatákat. Azt mondjuk, hogy a $\Phi = [A, X, \delta, Q, F]$ automata a Ξ_1 és Ξ_2 automatákból **szorzat konstrukcióval** keletkezik, ha

1. $A = A_1 \times A_2$
2. $Q = Q_1 \times Q_2$
3. Bármely $a \in A$ állapotra és $x \in X$ bemenő jelre $\delta(\Phi, a, x) = \delta(\Xi_1, a_1, x) \times \delta(\Xi_2, a_2, x)$.

A Φ automata kezdőállapot halmaza által generált részautomatáját a Ξ_1 és Ξ_2 automaták **szorzat-automatájának** nevezzük és $\Xi_1 \times \Xi_2$ -vel jelöljük.

3.1.2 Propozíció

Az $A_1 \times A_2$ minden $B_1 \times B_2$ részhalmazára, minden $x \in X$ bemenő jelre és minden $w \in X^*$ bemenő szóra

1. $\Delta(\Xi_1 \times \Xi_2, B_1 \times B_2, x) = \Delta(\Xi_1, B_1, x) \times \Delta(\Xi_2, B_2, x)$, és
2. $\Delta(\Xi_1 \times \Xi_2, Q_1 \times Q_2, w) = \Delta(\Xi_1, Q_1, w) \times \Delta(\Xi_2, Q_2, w)$

3.1.3 Definíció

Tegyük fel, hogy a $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ és a $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$ automaták állapot halmazai diszjunktak. Azt mondjuk, hogy a $\Phi = [A, X, \delta, Q, F]$ automata a Ξ_1 és Ξ_2 automatákból **unió konstrukcióval keletkezik**, ha

1. $A = A_1 \cup A_2$
2. $Q = Q_1 \cup Q_2$
3. Bármely $a \in A$ állapotra és $x \in X$ bemenős jelre $\delta(\Phi, a, x) = \delta(\Xi_1, a, x)$ ha $a \in A_1$, és $\delta(\Phi, a, x) = \delta(\Xi_2, a, x)$ ha $a \in A_2$.

A Φ automata kezdőállapot halmaza által generált részautomatáját a Ξ_1 és Ξ_2 automaták **unió-automatájának nevezzük** és $\Xi_1 \cup \Xi_2$ -vel jelöljük.

3.1.4 Propozíció

Az A_i állaptohalmaz minden B_i részhalmazára ($i = 1, 2$), minden $x \in X$ bemenő jelre és minden $w \in X^*$ bemenő szóra

1. $\Delta(\Xi_1 \cup \Xi_2, B_1 \cup B_2, x) = \Delta(\Xi_1, B_1, x) \cup \Delta(\Xi_2, B_2, x)$,
2. $\Delta(\Xi_1 \cup \Xi_2, Q_1 \cup Q_2, w) = \Delta(\Xi_1, Q_1, w) \cup \Delta(\Xi_2, Q_2, w)$

3.1.5 Definíció

Azt mondjuk, hogy a $\Phi = [B, X, \delta, Q, G]$ automata a $\Xi = [A, X, \delta, Q, F]$ automatából **részhalmaz konstrukcióval keletkezik**, ha

1. $B = 2^A$
2. Minden $b \in B$ állapotra és $x \in X$ bemenő jelre $\delta(\Phi, b, x) = \Delta(\Xi, b, x)$.

A Φ automata kezdőállapot halmaza által generált részautomatáját a Ξ automata **részhalmaz-automatájának** nevezzük és $2^\wedge \Xi$ -vel jelöljük.

3.1.6 Definíció

Azt mondjuk, hogy a $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ és a $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$ automaták **izomorfak**, ha létezik olyan $\alpha : A_1 \to A_2$ bijekció, hogy minden $a \in A_1$ állapotra és $x \in X$ bemenő jelre $\delta(\Xi_1, a, x)\,\alpha = \delta(\Xi_2, a\,\alpha, x)$.

3.1.7 Tétel

Bármely Ξ_1 és Ξ_2 automatára 2^($\Xi_1 \times \Xi_2$) izomorf a 2^($\Xi_1 \cup \Xi_2$) automatával, vagyis a szorzat-automata részhalmaz-automatája izomorf az unió-automata részhalmaz-automatájával.

3.1.8 Következmény

Determinisztikus Ξ_1 és Ξ_2 automatára a szorzat-automata izomorf az unió-automata részhalmaz-automatájával.

3.2 Irányítható automaták

3.2.1 Definíció

Azt mondjuk, hogy a w bemenő szó **összefésüli** a Ξ determinisztikus automata a és b állapotait, ha Δ (Ξ ,a,w)= Δ (Ξ ,b,w). Ha az *a* és *b* állapotokra létezik őket összefésülő szó, akkor a-t és b-t **összefésülhetőnek** nevezzük. Végül, ha a *w* bemenő szó az összes állapotpárt összefésüli, akkor w-t **irányító szónak** nevezzük. Magát a Ξ automatát pedig **irányíthatónak** mondjuk, ha létezik irányító szava.

3.2.2 Definíció

Tekintsük a Ξ automata különböző a és b állapotait és legyen

$$T(a,b)=\{p|\, p \in X^*, \text{ p összefésüli a-t és b-t, és p hossza minimális}\}.$$

Képezzünk ezekután egy olyan két oszlopos táblázatot, melynek első oszlopába azokat az [a,b] párokat írjuk, amelyekre T(a,b) nem üres, az [a,b]-val jelölt sor második oszlopába pedig magát T(a,b)-t írjuk. A keletkező táblázatot (ami megfelel a Maple tábla adatstruktúrájának) jelöljük $DirTab_{\Xi}$ -vel, és nevezzük a Ξ automata **irántyó táblájának**. Az [a,b]-vel jelölt sor második oszlopára a

Maple szintaxisát követve $DirTab_{\Xi}$ [a,b]-val hivatkozunk.

3.2.3 Propozíció

n állapotú irányítható automata egy irányító szava az irányító tábla ismeretében O(n) idő alatt előállítható.

3.2.4 Propozíció

Kommutatív automata esetén az automata az irányító táblájának ismeretében előállítható minimális hosszúságú irányító szó.

Imreh és Steinby 1955-ben publikált egy O(n^2 *m) időbonyolultságú algoritmust determinisztikus automaták irányíthatóságának eldöntésére. Az algoritmus alapötlete az alábbi észrevételeken alapul.

Legyen Ξ automata és jelölje $\mu(k, \Xi)$ azon (a,b) párok halmazát, mely párok legfeljebb k hosszuságú bemenő szóval összefésülhetők. A definíció alapján világos, hogy a $\mu(0, \Xi)$, $\mu(1, \Xi)$, $\mu(2, \Xi)$, ... halmazok növekvő sorozatot alkotnak, vagyis minden k természetes számra $\mu(k, \Xi) \subseteq \mu(k+1, \Xi)$. Továbbá, nem nehéz belátni, hogy az a és b állapototkat az xp szó akkor és csak akkor fésüli össze, ha a $\delta(a, x)$ és a $\delta(b, x)$ állapotokat a p szó összefésüli. Ez alapján a $\mu(k, \Xi)$ kiszámítására az alábbi rekurzív formula adódik:

1. $\mu(0, \Xi) = \{(a,a) \mid a \in \Xi_1\}$,
2. $\mu(k+1, \Xi) = \mu(k, \Xi)$ union $\{(a,b) \mid$ létezik $x \in \Xi_2$, amelyre ($\delta(\Xi, a, x), \delta(\Xi, b, x)$) eleme $\mu(k, \Xi)$ -nek $\}$

Végül, ha valamely k-ra $\mu(k, \Xi) = \mu(k+1, \Xi)$, akkor $\mu(k, \Xi) = \mu(\Xi)$, ahol $\mu(\Xi)$ az összefésülhető (a,b) párok halmaza.

Az eddig elmodottak szerint a Ξ automata irányítható voltának eldöntésére az alábbi eljárás adódik.

1. Állítsuk elő rendre a $\mu(0, \Xi)$, $\mu(1, \Xi)$, $\mu(2, \Xi)$, ... halmazokat mindaddig, amíg $\mu(k, \Xi) = \mu(k+1, \Xi)$ legelőször teljesül.
2. Ha ekkor $\mu(k, \Xi)$ (= $\mu(\Xi)$) az összes (a, b) állapotpárt tartalmazza, akkor Ξ irányítható, különben pedig nem.

Imreh és Steinby cikkükben egy nxn-es logikai mátrixot használnak annak jelzésére, hogy az állapothalmaz i-dik és j-dik eleme

összefésülhető-e. Célszerű azonban ehelyett egy bővebb információt hordozó adatstruktúrát is választani, a Ξ automata irányító tábláját, ami alkalmas a logikai mátrix szerepének betötésére, miközben használata nem növeli az algoritmus időbonyolultságát. Az így keletkezett eljárást nevezzük **módosított Imreh-Steinby algoritmusnak**.

3.2.5 Propozíció

A módosított Imreh-Steinby algoritmus időbonyolultsága O($m\,n^2$).

3.2.6 Propozíció

A módosított Imreh-Steinby algoritmus irányítható automata esetén előállítja az automata irányító tábláját.

3.2.7 Tétel

A módosított Imreh-Steinby algoritmus O($m\,n^2$) időkomplexitással alkalmas az irányíthatóság eldöntésére és irányító szó előállítására.

3.3 μ-automaták

3.3.1 Definíció

Legyen Ξ automata, legyen továbbá

$$W(\Xi)=\{w \mid w \in X^*, \text{létezik olyan } a, b \in \Xi_1 \text{, amelyre w eleme a } DirTab_{\Xi}[a,b] \text{ halmaznak}\}.$$

Legyen továbbá Y ábécé, amelyre a $|Y|=|W(\Xi)|$ és $\phi : Y \to W(\Xi)$ kölcsönösen egyértelmű leképezés. Azt a $\mu(\Xi) = [\Xi_1, Y, \delta, \Xi_4, \Xi_5]$ automatát, amelyre a $\delta(\mu(\Xi), a, y) = \delta(\Xi, a, y\phi)$ egyenlőség minden a állapotra és $y \in Y$ bemenő jelre teljesül a Ξ automatához tartozó **μ-automatának** nevezzük.

3.3.2 Tétel

A Ξ automata akkor és csak akkor irányítható, ha $\mu(\Xi)$ létezik és irányítható.

3.4 Reprezentációk

A négyes szabályként megfogalmazott didaktikai elv a NCTM 2000 standardok kötött látott napvilágot, és azóta számos didaktikai témájú könyvben is publikációban hivatkoznak rá. Az elv gyakorlati alkalmazása többségében a kalkulus területén valósult meg, sőt bizonyos szerzők eleve a kalkulusra korlátozzák az elv megfogalmazását.

A függvénytan valóban kellemes terület. A függvény helyettesítési értékeinek egy sorozata példát ad numerikus reprezentációra, míg a függvény formulával való megadása illetve grafikonja a szimbolikus ill. a grafikus reprezentációt adja. A függvénytulajdonságok természetes nyelven való megfogalmazása pedig biztosítja a leíró, verbális reprezentáció megjelenését.

A négyes szabály megfogalmazása

"every topic should be presented numerically, graphically, symbolically and verbally"

azt sugallja, hogy a matematikai problémákat (mindegyiket!) négy különféle módon, egy leíró, egy szimbolikus, egy grafikus és egy numerikus reprezentációval célszerű, sőt melegen ajánlott reprezentálni.

Ugyanakkor egy adott matematikai fogalmat, problémát vagy jelenséget a négyes szabályban felsorolt reprezentációk bármelyikével általában többféle módon is reprezentálhatunk. Tehát az egyes reprezentációk nem egyértelműen meghatározottak, sőt sokszor célszerű több különböző grafikus vagy éppen több különböző szimbolikus reprezentációval segíteni a vizsgált jelenség megértését. Látható tehát, hogy a négyes szabályban szereplő fogalmak (numerikus, grafikus, szimbolikus és verbális) valójában nem egy-egy reprezentációt, hanem reprezentáció típusokat határoznak meg. A konkrét reprezentációk e típusok különféle megjelenési formái.

Az egyes reprezentációs típusok sok esetben többszörösen jelennek meg egy probléma vizsgálata során. Ennek megfelelően szó sincs négy különböző reprezentációról, legfeljebb négy különböző reprezentációs típusba tartozó reprezentációk kellő számú, néha csak egy-kettő, néha akár öt-hat alkalmazásáról. Ennek megfelelően célszerűbbnek látjuk a négyes szabály helyett a többszörös reprezentáció elvét használni, melyet az alábbi formában fogalmazhatunk meg. **A matematikai problémákat, fogalmakat, jelenségeket ajánlatos több, a vizsgálat tárgyára nézve realisztikus reprezentációval szemlélhetni.**

Ezek a reprezentációk a konkretizált, a grafikus, a szimbolikus és a verbális reprezentáció típusokból kerülhetnek ki. A konkretizált reprezentáció típus bevezetésével **a négyes szabály általánosításához** jutunk, míg a realisztikus reprezentációkra való szorítkozás biztosítja, hogy a tárgyalás során felmerülő reprezentációk a vizsgált matematikai fogalom, probléma, vagy jelenség szempontjából didaktikai ésszerűséggel bírjanak, és ily módon járuljanak hozzá a jobb megértéshez és a tárgyalás nagyobb kognitív hatékonyságához.

3.5 Az automat elméleti fogalmak tanítása

A véges állapotú gépek matematikai koncepciójának didaktikai megközelítésen alapuló tárgyalásánál a **való világ példáiból** indulunk ki (Archimédeszi elv). A véges állapotú gépek intuitív fogalmát használjuk arra, hogy bevezessük és implementáljuk a véges automaták két reprezentációját: az átmeneti táblát és az átmenet mátrixot. Ebben a fázisban a Maple automata elméleti csomagjának számos eljárását használjuk, ám ezek mindegyikét **fekete doboznak** (black box) tekintjük. Nem törődünk ugyanis azzal, hogy az egyes eljárások hogyan dolgoznak. Elegendő tisztában lenni felhasználásukkal, mivel ezeket az eljárásokat jelen tárgyalási szakaszban kísérleti eszközökként használjuk.

A didaktikai cél az **absztrakció folyamatának megvilágítása**. A matematikai modell megalkotása során olyan tisztán matematikai eszközöket kell találnunk, amelyek alkalmasak arra, hogy leírják a szekvenciális gépek összetevőit és működését. A munkát **tapasztalatgyűjtéssel** kezdjük. Az automata elméleti csomagot használjuk arra, hogy tapasztalatokat gyűjtünk az általánosított átmenetfüggvény működéséről. Kísérleteink rávezetnek bennünket arra, hogy az összetett függvény képzése alkalmas eszköz olyan valóságos dolgok modellezésére, mint a bementi szalag és az olvasó fej. Földerítjük az általánosított átmenetfüggvény legfontosabb tulajdonságait, mely alapján képesek leszünk formálisan definiálni a determinisztikus automata és az általánosított átmenetfüggvény fogalmát, valamint a szavak és nyelvek felismerését. A tanulási **folyamat végére teljes egészében megértjük az implementáció működését, mely révén az eddig fekete dobozként működő eljárások fehér dobozzá vállnak** (black box - white box).

Ezek után folytatjuk az absztrakció folyamatát. Az absztrakció első lépésével szekvenciális gépek intuitív fogalmára alapozva megalkottuk a determinisztikus automata fogalmát. Az absztrakció következő lépésében erre a determinisztikus modellre alapozva felépítjük a nemdeterminisztikus automata fogalmát . Ugyanúgy, mint korábban, most is az **induktív eljárást** használjuk: kísérletezés, az eredmények és megfigyelések formalizálása, általánosítás és az új koncepció bevezetése.

A kiinduló pontunk a determinisztikus automata átmenet gráfja, amely a determinisztikus viselkedésnek köszönhetően, bizonyos speciális tulajdonságokkal rendelkezik. Azt kérdezzük, hogy mi történne, ha elvetnénk ezeket a tulajdonságokat, és megpróbálnánk minden egyes irányított címkézett gráfot valamilyen képzeletbeli automata átmenetfüggvényének tekinteni? Teljes egészében leírjuk ennek a feltevésnek a következményeit. Először azt kell realizálnunk, hogy megváltozik az átmenetfüggvény koncepciója. Ez a változás ugyanakkor azzal jár, hogy olyan új fogalmakat kell bevezetnünk: lehetséges futások halmazát futás helyett, lehetséges állapotok halmazát az aktuális állapot helyett. Fölfedezünk egy a bemenő szó feldolgozását leíró újabb reprezentációt, amelyet a lehetséges futások fájának nevezünk. **Hasonlóan a determinisztikus esethez a kísérleteink végére mindent tudunk ahhoz, hogy formalizáljuk a nem determinisztikus automata definícióját.** Didaktikai szempontból nézve tehát a nem determinisztikus általánosított átmenetfüggvény fekete dobozából fehér dobozt készítünk.

3.6 Az automata elméleti problémamegoldás tanítása

A dolgozatban mintaként két hasonló karakterű feladatot oldunk meg. Mindkét feladatban egy-egy automatát kell találnunk, ami felismer egy-egy előre adott nyelvet. A feladatmegoldás során a papír-ceruzával való munka során előszeretettel alkalmazott adaptív módszert használjuk. Elindultunk egy olyan automatáról, amely felismert egy a megadotthoz közelálló, ahhoz "hasonló" nyelvet, majd több lépésben kisebb nagyobb változtatásokat végeztünk a keletkező automatákon. Tesszük ezt mindaddig, amíg nem eljutottunk a feladat megoldását szolgáltató automatához. Ez az eljárás végeredményét tekintve nem teljes automatához vezetett az első esetben, és nemdeterminisztikus automatához a második feladata megoldása során.

Általában azt mondhatjuk, hogy az esetek többségében a természetes emberi gondolkodás az automata elméleti feladatok megoldása során nemdeterminisztikus illetve nem teljes automatához vezet. Természetes viselkedés, hogy a feladat megoldása szempontjából irreleváns állapotokat és átmeneteket már nem tüntetjük fel az automata átmeneti gráfján. Ennek következtében a keletkező automata nem lesz teljes. Ugyanakkor az irreleváns állapotok és az irreleváns átmenetek nem segítenek a keletkező automata működésének megértésében, amely azt jelenti, hogy a teljes és determinisztikus automaták kognitív hatékonysága alacsonyabb.

Ami a determiniszmust illeti, azt mondhatjuk, hogy a tanulók általában előnyben részesítik a szemléletesebb, ikonikus síkon ható reprezentációkat. Automaták esetén ez nem más, mint az átmenet gráf. Ebben a reprezentációban pedig az automata véletlen választásokon alapuló működés modellje természetes egyszerűséggel kezelhető. A tanulóknak ugyanis nem kell mást tenniük, mint utakat keresni az átmenet gráfban. Ezzel szemben a determinisztikus eset tárgyalása szimbolikusan, algebrai eszközökkel is megfelelő kognitív hatékonysággal kezelhető. Megállapíthatjuk tehát, hogy egyrészt az automata működésének különböző modelljeihez az egyes reprezentációkban jelentősen különböző kognitív hatékonyság társul, másrészt a grafikus reprezentációra alapuló problémamegoldás oda vezet, hogy a tanulók kialakulóban lévő kognitív struktúrája sokkal közelebb esik a nemdeterminisztikus modellhez, mind a véges állapotú gépek determinisztikus modelljéhez.

3.7 A Maple munkalapok didaktikai strukturája

A tanulás időben zajló folyamat megállásokkal, visszacsatolásokkal, visszalépésekkel, kísérleti fázisokkal tarkítva. **A kísérletek alapvető szerepet játszanak a megfigyelés során a tapasztalatok összegyűjtésében.** Az itt felmerülő legnehezebb kérdés az, hogy mennyi időt kell töltenünk kísérletezéssel? Más szavakkal azt is kérdezhetnénk, hogy **hányszor kell egy kísérletet megismételni** annak érdekében, hogy az osztályban lévő minden egyes tanuló képes legyen az eredmények felvázolására, a tapasztalt jelenségek mögöttes meghúzódó általános szabályok megsejtésére és azok formalizálására? A nehézség itt abban áll, hogy az osztály tanulóinak felfogó képessége jelentősen eltéréseket mutathat.

A válaszunk erre a kihívásra a CAS használata. A Maple, mint bármelyik más interaktív CAS, lehetőséget ad a felhasználónak

utasítások adott sorozatának többszöri, ismételt végrehajtására, miközben a felhasználó a számításban résztvevő adatok módosítását is könnyedén elvégezheti. Mindössze a Maple munkalapok szerkezeti struktúrálása **mellett be kell vezetnünk a munkalapok didaktika strukturálását**. A tanulók figyelmét fel kell hívni arra, hogy melyik a kezdőpontja és hol a végpontja azoknak egymást követő és logikailag összefüggő Maple utasításoknak, amelyek alkalmasak egy jelenség valamilyen szempontú megfigyelésére, vagyis amelyek **kísérletet alkotnak**. Ha a Maple eljárásainkat és utasításainkat úgy szervezzük, hogy a kísérlet első utasítása azokat az adatokat tartalmazza, amelyektől az adott kísérlet függ, akkor ez az első utasítás egy **PoP** (Point of Practice) helyet definiál a tanulási eljárás folyamatában. A PoP pontot tehát arra használjuk, hogy megindítsunk vele egy kísérletet és segítsünk a tanulóknak abban, hogy megtalálja azokat az adatokat, amelyeket megváltoztathat a kísérletezés során. A kísérletet alkotó utasítások sorozatának utolsó parancsát **EoP**-vel (End of Practice) jelöljük.

Ily módon tehát a tanítási anyagunk számos PoP-EoP párt tartalmazhat, amelynek mindegyike egy-egy kísérlet első illetve utolsó utasítását jelöli ki. A kísérlet természetesen nem csak Maple utasításokat és azok outputjait, hanem szöveges magyarázatokat, értelmezéseket is tartalmazhat. Nem tűnik indokoltnak a PoP-EoP párok egymásba skatulyázása, az átfedések pedig egyenen zavaróak. **Eszerint azt is mondhatjuk, hogy Maple munkalapunk didaktikai struktúráját idegen PoP-EoP párok alkotják.** A kísérlet input adatait a POP utasítás tartalmazza. A kísérlethez tartozó utasítások sorozatát a tanulók annyiszor hajtják végre, ahányszor szükségük van a kísérlet eredményeinek megértéséhez és az eredmények helyes interpretálásához.

4 Konklúzió

Történetileg a természetes útja a véges automata bevezetésének a determinisztikus automata matematikai modelljének megalkotásával kezdődik. A definíció és annak taglalása erősen algebrai beállítottságú az automata fogalom szimbolikus reprezentációján alapul. A tanuló tehát ezek alapján megtanulja, hogy hogyan kezelje a determinisztikus automatát algebrai eszközökkel, hogyan tudja leírni annak működését, hogyan használja az automatákat szavak felismerésére és nyelvek elfogadására. Ezek után a nem determinisztikus automata bevezetése a determinisztikus automata fogalom általánosításaként történik. Ez a kezelés a szimbolikus reprezentáción épül, amely rendkívül nehézzé teszi a nem determinisztikus automata működésének megértését. A nem determinisztikus automaták tanítása valóban a komoly didaktikai kihívást jelent minden előadó számára.

Ugyanakkor azt láttuk, hogyha a grafikus reprezentációt választjuk a szimbolikus helyett, akkor a nem determinisztikus automata könnyűvé és egyszerűen kezelhetővé válik. Valóban a nem determinisztikus automata az intuitív emberi gondolkodáshoz közelebb állónak bizonyul. Másrészt a nem determinisztikus automata exponenciálisan tömörebb, mint a determinisztikus, amely azt jelenti, hogy exponenciálisan kevesebb állapotot illetve átmenetet tartalmaz. Ez az egyszerűbb belső struktúra megengedi, hogy egyszerűbb, és világosabb kezelését adjuk a különböző állítások matematikai bizonyításának.

1 Preliminaries

The main feature of Maple architecture is that except for its Kernel which was developed in C, all Maple procedures was developed in Maple's own language, and most of them are located in different user defined packages. The `aut` package is a collection of Maple procedures to create, manipulate, and investigate finite automata, acceptable languages and regular expressions. The package introduces several new types like `automaton` for finite automata, `deterministic` for deterministic automata, `regular` for regular expressions and others. The user is also provided with tools to work with these types of objects including the controlled and random establishment of automata and/or execution of changes on their certain properties. For example we can easily add or delete states, input signals or even transitions. The `aut` package offers more than thirty-five procedures and covers essentially the results of classical automata theory from different automata constructions thought parsing and simplifying regular expressions to Kleene`s theorem, the analysis and synthesis of finite automata. The package which consists of more than 2500 source lines offers 18 procedures for creating, manipulating and visualizing automata. Handling regular expression is aided by 7 procedures, while further 8 procedure provide tools for different automata constructions.

2 Goals and realization

The main goal of this dissertation is to find answer for two basic questions

1. *How can the aut package be used in automata theoretic research?*

2. *Can aut package be used as tool for teaching the results of automat theory, and which didactic methods can help the teachers to effectively teach the notions and theorems of automata theory, and the students to learn automata theoretic problem solving.*

according to these goals the dissertation consists of three parts. First part is an introduction to the usage of aut package. The second part contains some theoretic results associated with the usage of the package. In the end the last part is didactic oriented, where we work out didactic methods for the effective usage of `aut` package in teaching automata theory.

The detailed discussion of the package is impossible in the frame of a dissertation. For that reason we **concentrate on the usage** of the package while we also introduce the basic notions and notations. We deal with the implementation of different notions, review the different automata constructions (subset construction, product construction, etc.), next we describe the algorithm of the `Compose` procedure, the construction engine of the package.

In the second part of the dissertation we prove some automata theoretic result related by the usage of the package. First we prove a general theorem on composition of automata constructions. Next we turn to the discussion of **Imreh-Steinby algorithm** and show that by using the **merging table** instead of logical matrix the algorithm becomes capable to produce directing word within the same time complexity.

The merging table of automata gives us the possibility to introduce an associated automaton for directable automata called μ-automaton. We show that the **directablity of the associated μ-automaton is necessary and sufficient condition of the directablity of automaton**. μ-automaton possess more input signal but much shorter directing words. It is open problem which role could μ-automaton play in the investigation of directability.

The last part of the dissertation deals with the didactic questions of the usage of aut package in the education. We analyze in details and illustrate **Rule of Four** which was published among NCTM 2000 standards. We point out the multiplicity of representation and the role of realistic representation in cognitive efficiency. In the end we propose a more general rule which we call the Rule of Multiple Representations.

In the end as an application of the reviewed didactic rules we work out two Maple worksheets. One **for teaching the notion of automata theory**, another one for **teaching automata theoretic problem solving**. Besides using the well known didactic principles, like principle of activity, principle of guided experimental learning etc., we work out and give examples for the didactic inner structure of Maple worksheets.

3 Theses

3.1 Composition of automata constructions

3.1.1 Definition

Consider automata $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ and $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$. We say automaton $\Phi = [A, X, \delta, Q, F]$ is created from automata Ξ_1 and Ξ_2 by **product construction** if

1. $A = A_1 \times A_2$
2. $Q = Q_1 \times Q_2$

3. For every state $a \in A$ and input signal $x \in X$ the equality $\delta(\Phi, a, x) = \delta(\Xi_1, a_1, x) \times \delta(\Xi_2, a_2, x)$ holds.

The subautomaton generated by the initial states of Φ is called the **product automaton** of automata Ξ_1 and Ξ_2 and is denoted by $\Xi_1 \times \Xi_2$.

3.1.2 Proposition

For every subset $B_1 \times B_2$ of $A_1 \times A_2$, for every signal $x \in X$ and input word $w \in X*$

1. $\Delta(\Xi_1 \times \Xi_2, B_1 \times B_2, x) = \Delta(\Xi_1, B_1, x) \times \Delta(\Xi_2, B_2, x)$, and
2. $\Delta(\Xi_1 \times \Xi_2, Q_1 \times Q_2, w) = \Delta(\Xi_1, Q_1, w) \times \Delta(\Xi_2, Q_2, w)$

3.1.3 Definition

Suppose the set of states of automata $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ and $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$ are disjoint. We say automaton $\Phi = [A, X, \delta, Q, F]$ is created from automata Ξ_1 and Ξ_2 by **union construction** if

1. $A = A_1 \cup A_2$
2. $Q = Q_1 \cup Q_2$
3. For every state $a \in A$ and input signal $x \in X$ $\delta(\Phi, a, x) = \delta(\Xi_1, a, x)$ if $a \in A_1$ and $\delta(\Phi, a, x) = \delta(\Xi_2, a, x)$ if $a \in A_2$.

The subautomaton generated by the initial states of Φ is called the **union automaton** of automata Ξ_1 and Ξ_2 and is denoted $\Xi_1 \cup \Xi_2$.

3.1.4 Proposition

For every subset B_i of A_i ($i = 1, 2$), for every input signal $x \in X$ and input word $w \in X*$

1. $\Delta(\Xi_1 \cup \Xi_2, B_1 \cup B_2, x) = \Delta(\Xi_1, B_1, x) \cup \Delta(\Xi_2, B_2, x)$,
2. $\Delta(\Xi_1 \cup \Xi_2, Q_1 \cup Q_2, w) = \Delta(\Xi_1, Q_1, w) \cup \Delta(\Xi_2, Q_2, w)$

3.1.5 Definition

We say automaton $\Phi = [B, X, \delta, Q, G]$ is created from automaton $\Xi = [A, X, \delta, Q, F]$ by using **subset construction** if

1. $B = 2^A$
2. For every state $b \in B$ and for every input signal $x \in X$ the equality $\delta(\Phi, b, x) = \Delta(\Xi, b, x)$ holds.

The subatomaton generated by the initial states of Φ is called the **subset automaton** of automata Ξ and is denoted by 2^ Ξ .

3.1.6 Definition

Automata $\Xi_1 = [A_1, X, \delta, Q_1, F_1]$ and $\Xi_2 = [A_2, X, \delta, Q_2, F_2]$ are called **isomorph** if there exists α : $A_1 \to A_2$ bijection for which for every state $a \in A_1$ and input signal $x \in X$ the equality $\delta(\Xi_1, a, x)\,\alpha = \delta(\Xi_2, a\,\alpha, x)$ holds.

3.1.7 Theorem

For every automata Ξ_1 and Ξ_2 the automaton 2^($\Xi_1 \times \Xi_2$) is isomorph with automaton 2^($\Xi_1 \cup \Xi_2$), i.e. the subset automaton of product automaton is isomorph with the subset automaton of product automaton.

3.1.8 Corollary

For deterministic automata Ξ_1 and Ξ_2 the product automaton is isomorph with the subset automaton of union automaton.

3.2 Directable automata

3.2.1 Definition

We say the input word w **merges** states a and b if Δ (Ξ ,a,w)= Δ (Ξ ,b,w). If there exists an input word which merges states a and b

then these states are called **mergable**. In the end if the input word w merges all pairs of states then w is called **directing word**. The automaton itself is called **directable** if there exists a directing word for it.

3.2.2 Definition

Consider two different states a and b of automaton Ξ and let

T(a,b)={ p | $p \in X^*$, p merges a-t and b-t, and the length of p is minimal }.

The **merging table** $DirTab_{\Xi}$ of automaton Ξ is table (in Maple sense) which indices are pairs [a,b] for which T(a,b) is nonvoid, while $DirTab_{\Xi}$[a,b]=T(a,b).

3.2.3 Proposition

One directing word of an n-state directable Automaton can be constructed from merging table in O(n) time.

3.2.4 Proposition

For commutative automata we can construct a minimal length directable word from the merging table of automaton.

Imreh and Steinby published an O(n^2 *m) algorithm for deciding the directability of automata. This algorithm is based on the following facts.

Consider automaton Ξ and denote the set of all pair (a, b) which can be merged by a word with length less than or equal to k by $\mu(k, \Xi)$. It is obvious that the sets $\mu(0, \Xi)$, $\mu(1, \Xi)$, $\mu(2, \Xi)$, ... form an increasing series, i.e. for every natural number k $\mu(k, \Xi) \subseteq \mu(k+1, \Xi)$. Furthermore it is easy to see, that the word xp merges states a and b if and only if the word p merges states $\delta(a, x)$ and $\delta(b, x)$. All this yields that the sets $\mu(k, \Xi)$ can be produced recursively by the following rules.

1. $\mu(0, \Xi)$ = {(a,a) | $a \in \Xi_1$ },
2. $\mu(k+1, \Xi)$ = $\mu(k, \Xi)$ union {(a,b) | there exists $x \in \Xi_2$, for which ($\delta(\Xi, a, x), \delta(\Xi, b, x)$) belongs to $\mu(k, \Xi)$ }

In the end if for some k the equality $\mu(k, \Xi) = \mu(k+1, \Xi)$ holds then $\mu(k, \Xi) = \mu(\Xi)$, where $\mu(\Xi)$ is the set of all mergable pairs.

We have obtained the following procedure for the decidability of automata.

1. Produce the set $\mu(0, \Xi)$, $\mu(1, \Xi)$, $\mu(2, \Xi)$, ... until $\mu(k, \Xi) = \mu(k+1, \Xi)$ holds.
2. Now if $\mu(k, \Xi)$ $(= \mu(\Xi)$) equals to the set of all pairs (a, b) then Ξ is directable, or else not.

Imreh and Steinby used a logical matrix to indicate that states i and j are mergable. It turned out, however, that one should choose a more informative data structure, namely the merging table of the automaton. We can do this without increasing the time complexity of algorithm. Let us call the resulting procedure the **modified Imreh-Steinby algorithm.**

3.2.5 Proposition

The time complexity of modified Imreh-Steinby algorithm is O($m\, n^2$).

3.2.6 Proposition

In case of directable automata the modified Imreh-Steinby algorithm produces the merging table of automaton.

3.2.7 Theorem

The modified Imreh-Steinby algorithm decides the directability of automaton and also produces the merging table in time O($m\, n^2$).

3.3 On μ -automata

3.3.1 Definition

Consider autonaton Ξ and let

W(Xi)={w | $w \in X^*$, there exist $a, b \in \Xi_1$, fro which w belongs to $DirTab_{\Xi}$ [a,b] }.

Consider alphabet Y for which |Y|=| $W(\Xi)$ | and let ϕ : $Y \to W(\Xi)$ bijection. Automaton $\mu(\Xi) = [\, \Xi_1, Y, \delta, \Xi_4, \Xi_5 \,]$ for which $\delta(\mu(\Xi), a, y) = \delta(\Xi, a, y\,\phi)$ holds for every state a and signal $y \in Y$ is called μ **-automaton** associated to autonaton a Ξ .

3.3.2 Theorem

Automaton Ξ is directable if and only if $\mu(\Xi)$ exists and is directable.

3.4 Representations

Rule of Four, as a basic didactic principle, was formulated among the NCTM 2000 standards and since then it has been quoted by numerous books and publications. Practically we can say it is accepted by the community of didactic experts. The usage of the Rule of Four, however, has been realized mainly in the field of calculus, in fact certain authors restrict the wording of the principle to the calculus itself.

Calculus is a pleasant field, indeed. A sequence of values of a function provides us with example for numeric representation, while the formula and the graph of the function illustrate symbolic and graphical representations, respectively. In the end by wording the basic features of the function on natural language we gain textual representation.

The wording of rule of four sounds as follows:

"Every topic should be presented numerically, graphically, symbolically and verbally."

This sentence evokes the impression that mathematical problems always have to be represented in four different manners. On the same time a given mathematical notion, problem or phenomenon can be represented several different ways, which yields that the different kinds of representations are far from being unique. As we will see later, in many cases it is practical to represent the investigated problem by more than one graphical or symbolic representation.

As all of the great and deep principles the Rule of Four is realized in an extremely simple form. The need of its application compels the user to rethink his/her notions concerning representations.

On the other hand in many cases the different representation types appear multiply in the course of the examination of a problem. According to this instead of speaking of four different representations, we can speak only of the usage of a certain number of representations belonging to four different types. This number varies problem to problem and can be only two, or as we have seen also five or six.

It seems to be more suitable to use the notation of **Rule of Multiple Representations** instead of Rule of Four. This principle can be formulated by using new notions introduced in this paper as follows. **Mathematical topics should be represented by multiple**

representations being rational to the subject of examinations. These representations can belong to the concretized, graphic, symbolic and verbal representation types.

By the introduction of concretized representation we obtain a generalization of Rule of Four, while the stipulation of using rational representations only provides us with the didactic rationality of representation arising during the discussion of mathematical notion, object or phenomenon. In this way the usage of representation like these contributes to better understanding and to a higher cognitive efficiency of discussion.

3.5 Teaching notions of automata theory

The basic notions of automata theory are suitable to illustrate the usage of didactic rules. When discussing the didactic approach for teaching the notion of automata theory we start from the real world, namely from intuitive notion of deterministic finite state machines (Archimedes Rule) which proves to be enough for the implementation and the introduction of two representations of finite automata: transition table and transition graph (Rule of Four). At this phase we use different procedures from Maple's automata theory package, but we use them as black boxes. We do not care how these procedures work. It is sufficient to be aware of their usage, as we want to use them only as experimental tools.

The didactic goal is **to highlight the process of abstraction**. In the course of building mathematical models we have to find pure mathematical tools which are suitable to describe the constituent parts and the operation of sequential machines. We use the existing implementation and gather experiences about how the generalized transition function works. We realize that the composition of function is the suitable means to model real world things like input tape and reading head. We explore all fundamental features of generalized transition function and this leads us to formulate the definition of deterministic automaton, the generalized transition function and the concept of word and language recognition. **By the end of this learning process we completely understand the behaviors of the implementation which becomes white box** from the initial black box.

Next we continue the process of abstraction. The first level abstraction built the concept of deterministic automaton from the intuitive notion of sequential machines. The second level abstraction is used to build the notion of nondeterministic automaton from deterministic one. As before we use the **inductive method** again: experiment, formulate the results of observations, generalize and introduce the new concepts.

Our starting point is the transition graph of deterministic automata, which have a certain special property due to determinism. We ask what would happen if we drop this property and tried to consider every directed-labeled graph as the transition function of an "imaginary automaton". We completely explore the consequences of this assumption. First we realize that the concept of transition function has to be changed. This change yields, however, that we have to introduce new notions such as the set of runs instead of run

or the current set of possible states instead of current state when processing an input word. We discover another useful representation of processing the input word called the tree of possible runs. Similarly to the deterministic case by the end of our experiments, we know everything to formulate the definition of (nondeterministic) automaton. **From didactic point of view we make the black box of nondeterministic generalized transition function to be white box.**

3.6 Teaching automata theoretic problem solving

As samples we solve two exercises of the same characteristic. In both cases we had to find automaton which recognizes a given language. We reached the solution by using the adaptive method. We begun with an automaton which recognized a language being related with the given language and we altered this automaton as long as we obtained the automaton required. This method led us to a **not complete** automaton in the first case and **nondeterministic automaton** in the second case.

In general we can assert that in most cases the **natural human thinking results in noncomplete and nondeterministic automata in the course of automata theoretic problem solving**. As for noncompletness the reason for this is that people are originally lazy. Yes, we do not like needless work and prefer to reach the result with the most smaller effort. On the same time irrelevant states and transition do not help us to comprehend the operation of resulting automata, which means the cognitive efficiency of complete and deterministic automata is lower.

As for nondeterminism we can say that in general students prefer to use graphic representations which is nothing else than the transition graph in case of automata. In this representation the operating model of automata based on casual choices can be handled on a natural way. The student has nothing else to do than to find different paths within the transition graph.On the same time the deterministic case can be handled cognitive effectively in symbolic representation on an algebraic way. So we can state that on the one hand different models of the operation of automata own different level of cognitive efficiency, on the other hand **the cognitive structure of students seems to be closer to nondeterministic model of finite state machines** than that of deterministic machines.

3.7 Didactic structure of Maple worksheets

Learning is a process, which runs in the course of time with stops, stepping back, feedback and experimental stages. **Experiments** play fundamental role in observation, in gathering experiences. The most difficult question is how much time has to be spent for a certain phenomenon to experiment with it. In other words **how much time has an experiment to be repeated**, so that every student in the class be able to draw up the results of experiment and then suspect and formulate the general rule behind it. Student's comprehension in a class may differ significantly.

Our answer for this challenge is based on the use of CAS. Maple, like every interactive CAS, allows users to repeatedly execute the

same sequence of commands, while user can change the value of any data taking part in the calculation. What we have to do is to introduce the didactic structure of Maple worksheets. We have to draw students' attention to the beginning and ending point of the sequence of consecutive Maple commands which constitutes an experiment. If we organize our Maple worksheet in such a way that the first command of the sequence always contains only the data which the experiment depends on then we obtain a **Point of Practice** (PoP) in the course of learning process. PoPs are used to indicate the beginning of an experiment and help student to find out which data have to be changed to obtain different output of the experiment. The last command of the experiment is called as end of PoP and denoted by EoP.

In this way out teaching material may contain several PoP-EoP pairs each of which designate the first and the last command of an experiment, whose input data can be found in the PoP command. An experiment can contain not only commands and their output but also textual clarifications and hints. It is not desirable to encapsulate PoP-EoP pairs, while overlapping is confusing. **Accordingly we can say that the didactic structure of Maple worksheets is constituted by disjoint PoP-EoP pairs.** The input data of experiment is located in PoP command. The sequence of commands can be repeated by the students as many time as desired. This number may very student by student.

4 Conclusion

Historically, the usual way for introduction the notion of finite automata is to begin with deterministic automata as the mathematical model of finite state machines. The definition and the discussion is highly algebraic based on symbolic representation of automata. First the students learn how to handle deterministic automata with algebraic tools how to describe their operation and how to use them to recognize words and languages. After that the notion of nondeterministic automaton is introduced as a generalization of deterministic case. This treatment is based on symbolic representation which makes the understanding the of nondeterministic automata very difficult. The teaching of nondeterministic automata means serious didactic challenge for all lecturer.

On the other hand we have seen that if we choose graphical representation instead of symbolic one then nondeterministic automata become to be easy and simple to handle. Indeed, nondeterminsitic automata turned out to be closer to intuitive human thinking. On the other hand nondeterminsitic automata are exponentially succinct than deterministic ones which means exponentially less states and less transitions. This simpler inner structure allows much simpler and clearer treatment of proofs of different statements.

5 Közlemények/Publications

5.1 Referált cikkek/Referred papers

„Rational representation of forests by tree automata”, Acta Cybernetica, 3 309-320 (1977)

„Didactic Approach for Teaching Nondeterminism in Automata Theory”, ZDM, Volume 35 (April), Number 2, 2003

„CAS based approach for discussing subset construction”, ZDM, Volume 35 (April), Number 2, 2003

„Determinism versus Nondeterminism”, ZDM, 2003

„Illustrated Analysis of Rule of Four using Maple”, TMCS, 2004

5.2 Egyetemi doktori disszertáció/Theses

„Faautomaták és erdők osztályai”, Egyetemi doktori disszertáció, Szegedi Egyetem, 1980

5.3 Könyvek/Books

„Maple 8 tételben”, Klincsik Mihállyal közösen, Novadat, 1996

„Bevezetés a Maple használatába”, André Heck „Introduction to Maple” c. Könyvének fordítása, Juhász Gyula Könyvkiadó, 1999

5.4 Egyéb közlemények/Other publications

„On implememting automata construction in Maple”, Proceedings of the International Symposium for 40th Anniversary of Pollak Mihály Collage of Engineering, 2002

„Repozitorikezelés Maple-ben I. ” CAS fórum, www.mathserv.hu, 2003

„Repozitorikezelés Maple-ben II.” CAS fórum, www.mathserv.hu, 2003

„Maple súgó oldalak készítése” CAS fórum, www.mathserv.hu, 2003.

5.5 Előadások/Lectures

„Automata constructions”, Előadás a Waterloo University Maple workshopján, 2002

„Investigating directable automata in Maple”, Automata elméleti konferencia, Debrecen, 2002

„Reguláris kifejezések kezelése Maple-ben”, JATE ITCS szeminárium, 2002

„A négyes szabály illusztrált elemzése”, Matematika didaktikai konferencia, Pécs, 2003