9725

Tomus 7. Fasciculus 3.

E Egyetemi Könyvtár SZEGED 14.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

FUNDAVIT: L. KALMÁR

REDIGIT: F. GÉCSEG

COMMISSIO REDACTORUM

A. ÁDÁM
M. ARATÓ
S. CSIBI
B. DÖMÖLKI
B. KREKÓ
Á. MAKAY
D. MUSZKA
ZS. NÁRAY
F. OBÁL
F. PAPP
A. PRÉKOPA
J. SZELEZSÁN
J. SZENTÁGOTHAI
S. SZÉKELY
J. SZÉP
L. VARGA
T. VÁMOS

SECRETARIUS COMMISSIONIS
J. CSIRIK

Szeged, 1986
Curat: Universitas Szegediensis de Attila József nominata

7. kötet 3. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

ALAPÍTOTTA: KALMÁR LÁSZLÓ

FŐSZERKESZTŐ: GÉCSEG FERENC

A SZERKESZTŐ BIZOTTSÁG TAGJAI

ÁDÁM ANDRÁS
ARATÓ MÁTYÁS
CSIBI SÁNDOR
DÖMÖLKI BÁLINT
KREKÓ BÉLA
MAKAY ÁRPÁD
MUSZKA DÁNIEL
NÁRAY ZSOLT
OBÁL FERENC
PAPP FERENC
PRÉKOPA ANDRÁS
SZELEZSÁN JÁNOS
SZENTÁGOTHAI JÁNOS
SZÉKELY SÁNDOR
SZÉP JENŐ
VARGA LÁSZLÓ
VÁMOS TIBOR

A SZERKESZTŐ BIZOTTSÁG TITKÁRA
CSIRIK JÁNOS

Szeged, 1986. január
A Szegedi József Attila Tudományegyetem gondozásában

The probabilistic behaviour of the *NFD* Bin Packing algorithm

By J. CSIRIK and E. MÁTÉ

Introduction

In the classical one-dimensional bin-packing problem we are given a list $L=(a_1, a_2, \ldots, a_n)$ of numbers (items) in the interval (0,1], which must be packed into a minimum number of unit-capacity bins (i.e. bins that can contain items totalling at most 1). It is well known that this problem is NP-hard [4], and accordingly a number of approximation algorithms have been developed for its solution. Johnson et al. analysed the best-known heuristics from a worst-case point of view [6]. Their analysis of approximation rules concentrated on the derivation of worst-case bounds of the form

$$A(L) \leqq \alpha \cdot OPT(L) + \beta$$

where α and β are constant and $A(L)$ and $OPT(L)$ are the numbers of bins required to pack L by algorithm A and an optimization rule, respectively. The multiplicative constant is an asymptotic bound on $A(L)/OPT(L)$ and it is the main focus of the analysis. The least constant gives the tight bound of the algorithm.

Two of the best-known heuristics are First Fit (*FF*) and Next Fit (*NF*). In *FF*, we place a_1 in bin 1, and treat the remaining items in order, placing each in the first bin that still has enough room for it (if no opened bin has enough room, then we start a new bin). In *NF* we similarly place a_1 in bin 1, and if a_i will fit into the last-opened bin, then we put it in this bin; otherwise, we start a new bin (which will be the last-opened bin). *FFD* (First Fit Decreasing) and *NFD* (Next Fit Decreasing) differ from *FF* and *NF* only in that the list is initially sorted so that

$$a_1 \geqq a_2 \geqq \ldots \geqq a_n.$$

Johnson proved that the tight asymptotic bound for *FF* is 17/10, for *FFD* is 11/9, and for *NF* is 2. Baker [1] showed that the sum

$$\gamma = \sum_{i=1}^{\infty} \frac{1}{b_i} = 1 + \frac{1}{2} + \frac{1}{6} + \ldots \approx 1.691$$

where $b_1=1$ and $b_{i+1}=b_i(b_i+1)$ for $i \geqq 1$, is a tight asymptotic bound for *NFD*.

Numerous results have been achieved on a second line of research: analysis of the expected behaviour of a heuristic algorithm. In this approach, one assumes a density function for the items and establishes probabilistic properties of the heuristic, such as their expected performance. In the special case, lists consist of items independently and uniformly distributed in the interval (0, 1]. Frederickson [3] showed that the *FFD* rule is asymptotically optimal. Recently, Bentley [2] proved the unexpected result that *FF* is also asymptotically optimal. Hofri [5] and Ong [7] showed that for *NF*

$$\mathbf{E}(NF) = 1/6+2n/3$$

in this case, where $\mathbf{E}(NF)$ denotes the expected number of bins for the *NF* rule.

Results

Let we now assume that the items of $L=(a_1, a_2, \ldots, a_n)$ are independently and uniformly distributed in the interval (0, 1]. We then have

Lemma 1.

$$\overline{\lim_{n\to\infty}} \frac{\mathbf{E}(NFD)}{n/2} \leqq 2\left(\frac{\pi^2}{6}-1\right).$$

Proof. Let us sort the elements of L so that

$$a_{i_1} \geqq a_{i_2} \geqq \ldots \geqq a_{i_n}.$$

Let the number of elements in the interval

$$\left(\frac{1}{l+1}, \frac{1}{l}\right] \quad l = 1, 2, \ldots$$

be k_l.

Let us define the sliced NFD_r ($SNFD_r$) algorithm as follows: we pack the elements $> \frac{1}{r}$ in accordance with the *NFD* rule; we then complete the last-opened bin so that it will have at most $(r-1)$ elements; and finally, from the remaining items, we always pack r together. Clearly, for all L:

$$SNFD_r(L) \geqq NFD(L)$$

and

$$\lim_{r\to\infty} SNFD_r(L) = NFD(L).$$

Let $K_r = k_r+k_{r+1}+\ldots$.

Then, for the packing of L by $SNFD_r$, we need at most

$$k_1+\frac{k_2}{2}+\ldots+\frac{k_{r-1}}{r-1}+\frac{K_r}{r}+r$$

bins. Hence, for the expected value of bins with the $SNFD_r$ (for list $L=(a_1, a_2, \ldots, a_n)$):

$$\mathbf{E}(SNFD_r(L)) \leqq \sum_{k_1+k_2+\ldots+k_{r-1}+K_r=n} P(k_1, k_2, \ldots, k_{r-1}, K_r) \cdot$$

$$\cdot \left(k_1+\frac{k_2}{2}+\ldots+\frac{k_{r-1}}{r-1}+\frac{K_r}{r}+r\right)$$

where

$$P(k_1, k_2, \ldots, k_{r-1}, K_r) =$$

$$= \frac{n!}{k_1!\, k_2! \ldots k_{r-1}!\, K_r!}\left(\frac{1}{2}\right)^{k_1}\left(\frac{1}{6}\right)^{k_2}\cdots\left(\frac{1}{(r-1)r}\right)^{k_{r-1}}\left(\frac{1}{r}\right)^{K_r}.$$

Then

$$\mathbf{E}(SNFD_r(L)) \leqq \sum_{i=1}^{r-1}\frac{k_i}{i}\sum_{k_1+k_2+\ldots+k_{r-1}+K_r=n} P(k_1, k_2, \ldots, k_{r-1}, K_r)+$$

$$+\frac{K_r}{r}\sum_{k_1+k_2+\ldots+k_{r-1}+K_r=n} P(k_1, k_2, \ldots, k_{r-1}, K_r)+r =$$

$$= n\sum_{i=1}^{r-1}\frac{1}{i}\frac{1}{i(i+1)}\sum_{\substack{k_1+k_2+\ldots+k_{r-1}+K_r=n\\ k_i>0}}\frac{(n-1)!}{k_1!\, k_2! \ldots k_{i-1}!\,(k_i-1)!\, k_{i+1}! \ldots k_{r-1}!\, K_r!}\cdot$$

$$\left(\frac{1}{2}\right)^{k_1}\left(\frac{1}{6}\right)^{k_2}\cdots\left(\frac{1}{i(i+1)}\right)^{k_i-1}\cdots\left(\frac{1}{r}\right)^{K_r}+n\frac{1}{r}\frac{1}{r}\sum_{\substack{k_1+k_2+\ldots+k_{r-1}+K_r=n\\ K_r>0}}$$

$$\frac{(n-1)!}{k_1!\ldots k_{r-1}!\,(K_r-1)!}\left(\frac{1}{2}\right)^{k_1}\left(\frac{1}{6}\right)^{k_2}\cdots\left(\frac{1}{(r-1)r}\right)^{k_{r-1}}\left(\frac{1}{r}\right)^{K_r-1}+r =$$

$$= n\left(\sum_{i=1}^{r-1}\frac{1}{i^2(i+1)}+\frac{1}{r^2}\right)+r.$$

And hence for a fixed r

$$\overline{\lim_{n\to\infty}}\frac{\mathbf{E}(SNFD_r)}{n/2} \leqq 2\left(\frac{\pi^2}{6}-1\right).$$

Thus $\overline{\lim_{n\to\infty}}\dfrac{\mathbf{E}(NFD)}{n/2} \leqq 2\left(\dfrac{\pi^2}{6}-1\right)$ which completes the proof of Lemma 1.

If we now fix r and pack the list $L=(a_1, a_2, \ldots, a_n)$ with the *NFD* rule so that we do not count the bins which contain elements of two different intervals of type $\left(\dfrac{1}{l+1}, \dfrac{1}{l}\right]$ or element $<\dfrac{1}{r}$, then

$$\mathbf{E}(NFD) \leqq \sum_{k_1+k_2+\ldots+k_{r-1}+K_r=n} P(k_1, k_2, \ldots, k_{r-1}, K_r) \cdot$$

$$\cdot\left(k_1-1+\frac{k_2}{2}-1+\ldots+\frac{k_{r-1}}{r-1}-1\right).$$

From this, in a similar way as in Lemma 1, we get

$$\mathbf{E}(NFD) \geqq n\left(\sum_{i=1}^{r}\frac{1}{i^2}-1+\frac{1}{r}\right)-(r-1).$$

Thus, for a fixed r:

$$\lim_{n\to\infty}\frac{\mathbf{E}(NFD)}{\frac{n}{2}} \geqq 2\left(\sum_{i=1}^{r-1}\frac{1}{i^2}-1\right)+\frac{2}{r}.$$

The right side is a monotonously increasing function of r and

$$\lim_{r\to\infty}\left(2\left(\sum_{i=1}^{r-1}\frac{1}{i^2}-1\right)+\frac{2}{r}\right)=2\left(\frac{\pi^2}{6}-1\right).$$

This leads to

Lemma 2.

$$\lim_{n\to\infty}\frac{\mathbf{E}(NFD)}{\frac{n}{2}} \geqq 2\left(\frac{\pi^2}{6}-1\right).$$

Then, from Lemma 1 and Lemma 2:

Theorem 1.

$$\lim_{n\to\infty}\frac{\mathbf{E}(NFD)}{\frac{n}{2}}=2\left(\frac{\pi^2}{6}-1\right)\approx 1{,}29.$$

Let we now assume that the items of $L=(a_1, a_2, \ldots, a_n)$ are independently and uniformly distributed in the interval $(0, \alpha]$ $(0<\alpha\leqq 1)$, and let A be an integer such that

$$\frac{1}{A+1}<\alpha\leqq\frac{1}{A}.$$

In this case, $\mathbf{E}(OPT(L))\geqq\frac{n\cdot\alpha}{2}$ and, in a totally similar way as for $\alpha=1$, we can get

Theorem 2.

$$\lim_{n\to\infty}\frac{\mathbf{E}(NFD)}{\frac{n\alpha}{2}}=\frac{2}{\alpha^2}\left(\sum_{i=A+1}^{\infty}\frac{1}{i^2}-\frac{1}{A}(1-\alpha)\right).$$

J. CSIRIK
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SZEGED
ARADI VÉRTANÚK TERE 1.
SZEGED
HUNGARY

E. MÁTÉ
KALMÁR LABORATORY
OF CYBERNETICS
ÁRPÁD TÉR 2
SZEGED,
HUNGARY

References

[1] BAKER, B. S., COFFMAN, E. G., A tight asymptotic bound for Next-Fit-Decreasing Bin-Packing, Siam J. Alg. Disc. Meth. 2 (1981), 147—152.
[2] BENTLEY, J. L., JOHNSON, D. S., LEIGHTON, F. T., MCGEOCH, C. C., MCGEOCH, L. A., Some unexpected Expected Behaviour Results for Bin Packing Proc. ACM Symp. on Theory of Computing, 1984, 279—288.
[3] FREDERICKSON, G. N., Probabilistic Analysis for simple One and Two-dimensional Bin Packing Algorithms, Information Processing Letters, 11 (1980), 156—161.
[4] GAREY, M. R., JOHNSON, D. S., Computers and Intractibility: A Guide to the Theory of NP-Completeness, Freemann, San Francisco, CA, 1979.
[5] HOFRI, M., A Probabilistic Analysis of the Next-Fit Bin Packing Algoritm, J. of Algorithms, 5 (1984), 547—556.
[6] JOHNSON, D. S., DEMERS, A., ULLMANN, J. D., GAREY, M. R., GRAHAM, R. L., Worst -Case Performance Bounds for Simple One-Dimensional Bin Packing Algorithms, SIAM J. Comp. 3 (1974), 299—325.
[7] ONG, H. L., MAGAZINE, M. J., WEE, T. S., Probabilistic Analysis of Bin Packing Heuristics Operations Research, 32 (1984), 983—998.

(Received Apr. 10, 1985)

Langages écrits par un code infinitaire. Théorème du défaut

By Do Long Van

1. Notations et définitions

Soit A un alphabet non-vide. On note A^* le monoïde libre engendrè par A, i.e. l'ensemble de tous les mots finis sur A, y compris le mot vide noté ε, muni de l'opèration de concaténation. La longueur d'un mot f de A^* est noté $|f|$, et pour tout n, $1 \leqq n \leqq |f|$, $f(n)$ désigne la n-ième lettre du mot f. L'ensemble des mots infinis sur A est noté A^ω. Chaque mot u de A^ω est de longueur $|u| = \omega = \mathrm{Card}\,\mathbf{N}$ et est un application $u = \mathbf{N}^+ \to A$ qu'on ècrit souvent sous la forme $u = u(1)u(2)\ldots$. On pose $A^\infty = A^* \cup A^\omega$ et on appelle *langage infinitaire* (resp. *finitaire, purement infinitaire*) toute partie X de A^∞ (resp. A^*, A^ω). Si $X \subseteqq A^*$, X^ω désigne l'ensemble des mots infinis de la forme $x_1 x_2 \ldots$ avec $x_i \in X$ $(i = 1, 2, \ldots)$. En particulier, pour $f \in A^*$, $\{f\}^* = ff\ldots$. Pour rendre plus clair, dans la suite on notera souvent par $f, g, h, \ldots$ les mots finis, par $u, v, w, \ldots$ les mots infinis, et par $\alpha, \beta, \gamma, \ldots$ les mots dont la longueur est finie ou infinie.

On munit A^∞ d'un produit prolongeant celui de A^* de la manière suivante :

$$\forall u \in A^\omega \forall \alpha \in A^\infty : \; u\alpha = u;$$

$$\forall f \in A^* \forall u \in A^\omega : \; (fu)(n) = \begin{cases} f(n) & \text{pour} \quad 1 \leqq n \leqq |f|, \\ u(n - |f|) & \text{pour} \quad n > |f|. \end{cases}$$

On vérifie sans peine que A^∞ est alors un monoïde.

Pour toute partie X de A^∞ on note $X_{fin} = X \cap A^*$, $X_{inf} = X \cap A^\omega$ et on définit:

$$\begin{cases} X^{(0)} = \{\varepsilon\}, \\ X^{(1)} = X, \\ X^{(k)} = X_{fin} X^{(k-1)}, \quad k \geqq 2. \end{cases}$$

Alors, pour $k \geqq 1$, $X^{(k)} = X_{fin}^k \cup X_{fin}^{k-1} X_{inf}$, et par conséquent chaque élément α de $X^{(k)}$ peut se présenter sous l'une des deux formes :

(i) $\alpha = x_1 \ldots x_k$ avec $x_i \in X_{fin}$ $(i = 1, \ldots, k)$;

(ii) $\alpha = x_1 \ldots x_k$ avec $x_i \in X_{fin}$ $(i = 1, \ldots, k-1)$, $x_k \in X_{inf}$.

Comme d'habitude on note X^* le sous-monoïde de A^∞ engendré par X et pose $X^+ = X^* - \{\varepsilon\}$. On a évidemment

$$X^+ = \bigcup_{k=1}^{\infty} X^{(k)}.$$

Une partie X de A^∞ est un *code infinitaire* sur A (cf. [3]) si chaque élément α de X^+ peut se présenter uniquement sous l'une des deux formes (i) et (ii) pour un certain k, *ou*, d'une façon équivalente, si pour tous $x_1 \ldots x_n \in X^{(n)}$, $x'_1 \ldots x'_m \in X^{(m)}$, l'égalité

$$x_1 \ldots x_n = x'_1 \ldots x'_m$$

implique $n=m$ et $x_i = x'_i$ $(i=1, \ldots, n)$. Dans la suite, sauf spécification contraire, le mot « code » désignera un code infinitaire.

On appele *quasi-libre* (cf [4]) tout sous-monoïde M de A^∞ engendré par un code. Le code qui engendre M est appelé la *base* de M. L'ensemble de tous les sous-monoïdes quasi-libres est noté QL.

Étant donné un sous-monoïde M de A^∞ nous introduisons sur M_{inf} une relation linéaire transitive, notée « $\prec$ », de la manière suivante :

$$u \prec v \Leftrightarrow \exists f \in (M_{fin} - \varepsilon) : v = fu.$$

Si $u \prec v$ nous disons que u *est contenu* dans v ou v *contient* u. Un élément u de M_{inf} est dit *maximal* s'il n'existe aucun élément v de M_{inf} tel que $u \prec v$. On dit que le sous-monoïde M satisfait à la *condition de maximalité* si tout élément non-maximal de M_{inf} est contenu dans un certain élément maximal de M_{inf}. On appelle *chaîne* toute suite croissante $u_1 \prec u_2 \prec \ldots$ d'éléments de M_{inf} ordonnée par « $\prec$ ». Une chaîne peut être finie ou infinie. On dit que le sous-monoïde M satisfait à la *condition de chaîne finie* si toute chaîne d'éléments de M_{inf} est finie. La condition de chaîne finie implique évidemment celle de maximalité, mais l'implication inverse est fausse ([6], Exemple 2).

Un ensemble générateur d'un monoïde M est *minimum* s'il est inclus dans tout ensemble générateur de M. L'ensemble générateur minimum d'un monoïde, s'il existe, est clairement unique.

On appelle *distincte* toute partie X de A^∞ telle que $X_{inf} \cap X^+_{fin} X_{inf} = \emptyset$. Clairement sont distinctes toute partie finitaire ainsi que tout code infinitaire.

2. Langages écrits par un code infinitaire

Étant donnée une classe C de codes on dit qu'un langage X est *écrit par un code de la classe* C ou *C-éctit par un code* s'il existe un $Y \in C$ tel que $X \subseteq Y^*$. Dans le cas ou C coïncide avec la classe de tous les codes on dit simplement que X est *écrit par un code*.

Tout langage finitaire est clairement écrit par un code tandis qu'il existe des langages infinitaires qui ne peuvent écrit par aucun code. L'exemple d'un tel langage est A^∞ pour A non-vide (cf [5], Corollaire 2). Nous caractérisons dans cette section les langages qui sont écrits par un code (code préfixe, code suffixe, code bipréfixe, code normal).

Rappelons que pour toutes parties X et Y de A^∞ on définit

$$Y^{-1}X = \{\alpha \in A^\infty \mid \exists \beta \in Y\colon (\beta\alpha \in X) \,\&\, (|\beta| = \omega \to \alpha = \varepsilon\},$$

$$XY^{-1} = \{\alpha \in A^\infty \mid \exists \beta \in Y\colon \alpha\beta \in X\}.$$

Maintenant, pour toute partie X de A^∞, on pose

$$LB(X) = X^{-1}X \cap XX^{-1};$$

$$UD(X) = X^{-1}X;$$

$$UG(X) = XX^{-1};$$

$$BU(X) = X^{-1}X \cup XX^{-1}.$$

Un sous-monoïde M de A^∞ est par définition *libérable* (resp. *unitaire à droite, unitaire à gauche, biunitaire*) ssi $LB(M) \subseteqq M$ (resp. $UD(M) \subseteqq M$, $UG(M) \subseteqq M$, $BU(M) \subseteqq M$). Notant par LB (resp. UD, UG, BU) la classe de tous les sous-monoïdes libérables (resp. unitaires à droite, unitaires à gauche, biunitaires) on a donc

$$M \in Z \Leftrightarrow Z(M) \subseteqq M \quad \text{pour} \quad Z \in \{LB, UD, UG, BU\}.$$

On vérifie sans peine que, pour toute famille $\{X_i \mid i \in I\}$ de parties de A^∞ et pour tout $Z \in \{LB, UD, UG, BU\}$, $Z(\bigcap_{i\in I} X_i) \subseteqq \bigcap_{i\in I} Z(X_i)$. Les classes de sous-monoïdes LB, UD, UG, BU sont donc fermées par intersection. Comme A^∞ est biunitaire et par conséquent unitaire à droite, unitaire à gauch, libérable, il existe donc, pour toute partie X de A^∞, un plus petit sous-monoïde libérable (unitaire à droite, unitaire à gauche, biunitaire) de A^∞ contenant X qu'on note par $\overline{LB}(X)$ (resp. $\overline{UD}(X)$, $\overline{UG}(X)$, $\overline{BU}(X)$).

On associe maintenant à chaque Z de $\{LB, UD, UG, BU\}$ et chaque partie X de A^∞ une suite croissante $M_n^Z(X)$ de sous-monoïdes de A^∞ ainsi définie :

$$M_0^Z(X) = X^*, \quad M_{n+1}^Z(X) = [Z(M_n^Z(X))]^*, \quad n \geqq 0.$$

Posons:

$$M^Z(X) = \bigcup_{n\geqq 0} M_n^Z(X).$$

Proposition 2.1. *Pour tout* $Z \in \{LB, UD, UG, BU\}$ *et pour toute* X *de* A^∞, *on a :*

$$M^Z(X) = \overline{Z}(X).$$

Preuve. On a $M_n^Z(X) \subseteqq \overline{Z}(X)$ pour tout n. En effet, ceci est évident pour $n=0$. Puis, si $M_n^Z(X) \subseteqq \overline{Z}(X)$, on a :

$$M_{n+1}^Z(X) = [Z(M_n^Z(X))]^* \subseteqq [Z(\overline{Z}(X))]^* \subseteqq [\overline{Z}(X)]^* = \overline{Z}(X)$$

car $\overline{Z}(X) \in Z$. Donc $M^Z(X) \subseteqq \overline{Z}(X)$. D'autre part, comme la suite $M_n^Z(X)$ est croissante, on a :

$$Z(M^Z(X)) = \bigcup_{n\geqq 0} Z(M_n^Z(X)) \subseteqq \bigcup_{n\geqq 0} [Z(M_n^Z(X))]^* = \bigcup_{n\geqq 0} M_{n+1}^Z(X) = M^Z(X),$$

ce qui montre que $M^Z(X) \in Z$. Par minimalité de $\overline{Z}(X)$, il en resulte $\overline{Z}(X) \subseteqq M^Z(X)$. Ainsi $M^Z(X) = \overline{Z}(X)$.

Maintenant, si un langage X est écrit par un code, alors, du fait que QL est fermée par intersection ([6], Corollaire 6), on peut parler du plus petit sous-monoïde quasi-libre contenant X que l'on note $\overline{QL}(X)$.

Le théorème suivant caractérise les langages qui sont écrits par un code :

Théorème 2.2. *Pour tout langage infinitaire X, les conditions suivantes sont équivalentes :*

(i) *X est écrit par un code ;*
(ii) *Il existe $\overline{QL}(X)$ et $\overline{QL}(X)=M^{LB}(X)$;*
(iii) *$M^{LB}(X)$ satisfait à la condition de chaîne finie ;*
(iv) *$M^{LB}(X)$ satisfait à la condition de maximalité ;*
(v) *$M^{LB}(X)$ possède un ensemble générateur minimum distinct.*

Preuve. (i)⇒(ii) : supposons que X soit écrit par un code. Comme il a été dit plus haut, il existe $\overline{QL}(X)$. $\overline{QL}(X)$ est alors libérable car tout sous-monoïde quasi-libre est libérable ([6], Proposition 2). Utilisant la Proposition 2.1 avec $Z=LB$ et la minimalité de $\overline{LB}(X)$ on a $M^{LB}(X)=\overline{LB}(X)\subseteq\overline{QL}(X)$. Mais alors, par le Corollaire 4 en [6], $M^{LB}(X)$ est quasi-libre. Par minimalité de $\overline{QL}(X)$ on en déduit $\overline{QL}(X)\subseteq M^{LB}(X)$. Ainsi $\overline{QL}(X)=M^{LB}(X)$.

(ii)⇒(iii) puisque tout sous-monoïde quasi-libre satisfait à la condition de chaîne finie ([6], Proposition 3).

(iii)⇔(iv)⇔(v) est immédiat du Corollaire 1 en [6] et du fait que $M^{LB}(X)$, par la Proposition 2.1, est libérable.

(v)⇒(i) : Supposons $M^{LB}(X)$ possède un ensemble générateur minimum distinct. En vertu du Théoréme 1 en [6], $M^{LB}(X)$ est quasi-libre. X est donc écrit par un code qui est la base de $M^{LB}(X)$.

Une partie X de A^∞ est *préfixe* (*suffixe*) si aucun mot de X n'est facteur gauche (resp. facteur droit) propre d'un mot de X. La partie X est *bipréfixe* si elle est à la fois préfixe et suffixe. Toute partie préfixe (suffixe, bipréfixe) $X\neq\{\varepsilon\}$ est un code appelé *code préfixe* (resp. *suffixe, bipréfixe*).

Un code X est *normal* si $X_{fin}^+X_{inf}\cap X_{fin}^\omega=\emptyset$. Sont normaux évedemment tout code finitaire ainsi que tout code préfixe.

Un sous-monoïde M de A^∞ est dit *régulier* si $M_{inf}\cap M_{fin}^\omega=\emptyset$. Tout sous-monoïde régulier satisfait à la condition de chaîne finie, mais la réciproque n'est pas vraie (cf. [6], Exemple 4 (suite)).

Si un langage X est écrit par un code préfixe (suffixe, bipréfixe, normal), alors, parce que la classe des sous-monoïdes engendrés par codes préfixes (suffixes, bipréfixes, normaux) est fermée par intersection, on peut parler du plus petit sous-monoïde de cette classe qui contient X. Celui-ci est noté $\overline{P}(X)$ (resp. $\overline{S}(X)$, $\overline{BP}(X)$, $\overline{N}(X)$).

Remarque. Pour toute partie X de A^* on a

$$\overline{LB}(X)=\overline{QL}(X)=\overline{N}(X)=\overline{L}(X);$$

$$\overline{UD}(X)=\overline{P}(X)=\overline{PF}(X);\quad \overline{UG}(X)=\overline{S}(X)=\overline{SF}(X);$$

$$\overline{BU}(X)=\overline{BP}(X)=\overline{BPF}(X),$$

où $\overline{L}(X)$ est le plus petit sous-monoïde libre contenant X ; $\overline{PF}(X)$ (resp. $\overline{SF}(X)$, $\overline{BPF}(X)$) est le plus petit sous-monoïde qui est engendré par un code finitaire préfixe (resp. suffixe, bipréfixe) et qui contient X.

Théorème 2.3. *Pour tout langage infinitaire X, les conditions suivantes sont équivalentes :*
(i) *X est écrit par un code normal ;*
(ii) *Il existe $\overline{N}(X)$ et $\overline{N}(X)=M^{LB}(X)$;*
(iii) *$M^{LB}(X)$ est régulier.*

Preuve. (i)⇒(ii) : si X est écrit par un code normal, alors il existe $\overline{N}(X)$ qui, par le Théorème 2 en [6], est libérable. Par la Proposition 2.1 et par minimalité de $\overline{LB}(X)$, $M^{LB}(X)=\overline{LB}(X)\subseteq\overline{N},(X)$ d'où, par le Corollaire 8 en [6], $M^{LB}(X)$ est aussi engendré par un code normal. Donc $\overline{N}(X)=M^{LB}(X)$.

(ii)⇒(iii) est évident.

(iii)⇒(i) : supposons $M^{LB}(X)$ régulier. Par la Proposition 2.1, $M^{LB}(X)$ est libérable. En vertu du Théorème 2 en [6], $M^{LB}(X)$ est engendré par un code normal. Donc X est écrit par un code normal qui est la base de $M^{LB}(X)$.

Théorème 2.4. *Pour tout langage infinitaire X, les conditions suivantes sont équivalentes :*
(i) *X est écrit par un code préfixe (bipréfixe, suffixe) ;*
(ii) *Il existe $\overline{P}(X)$ (resp. $\overline{BP}(X)$, $\overline{S}(X)$) et $\overline{P}(X)=M^{UD}(X)$ (resp. $\overline{BP}(X)=M^{BU}(X)$, $\overline{S}(X)=M^{UG}(X)$) ;*
(iii) *$M^{UD}(X)$ (resp. $M^{BU}(X)$) est régulier ;*
(iv) *$M^{UD}(X)$ (resp. $M^{BU}(X)$, $M^{UG}(X)$) satisfait à la condition de chaîne finie ;*
(v) *$M^{UD}(X)$ (resp. $M^{BU}(X)$, $M^{UG}(X)$) satisfait à la condition de maximalité ;*
(vi) *$M^{UD}(X)$ (resp. $M^{BU}(X)$, $M^{UG}(X)$) possède un ensemble générateur minimum distinct.*

Preuve. Nous ne traitons que le cas de codes préfixes.

(i)⇒(ii) : si X est écrit par un code préfixe, il existe $\overline{P}(X)$ qui, par le Théorème 3 en [6], est unitaire à droite. En vertu de la Proposition 2.1 et de la minimalité de $\overline{UD}(X)$, $M^{UD}(X)=\overline{UD}(X)\subseteq\overline{P}(X)$. Alors, par le Corollaire 10 en [6], $M^{UD}(X)$ est aussi engendré par un code préfixe. D'où $\overline{P}(X)=M^{UD}(X)$.

(ii)⇔(iii) est immédiat du Théorème 3 en [6].

(iii)⇔(iv)⇔(v)⇔(vi) résulte immédiatement du fait que $M^{UD}(X)$, par la Proposition 2.1, est unitaire à droite et du Corollaire 9 en [6].

(vi)⇒(i) : supposons que $M^{UD}(X)$ possède un ensemble générateur minimum distinct. Alors, étant unitaire à droite, $M^{UD}(X)$, par le Théorème 3 en [6], est engendré par un code préfixe. Par conséquent X est écrit par un code préfixe.

3. Théorème du défaut

On montre dans cette section que le théorème du défaut énoncé sous la forme du Théorème 3.2 en [1] est encore valide pour le cas des langages et codes infinitaires.

Nous avons besoin du lemme suivant :

Lemme 3.1. *Soit X un langage infinitaire écrit par un code et soit Y la base*

de $\overline{QL}(X)$. *Alors tout élément de* Y *est initiale et terminale d'au moins un mot dans* X *; c'est a dire que l'on a ;*

$$Y \subseteqq X(Y^*)^{-1} \cap (Y^*_{fin})^{-1} X.$$

Preuve. Démontrons $Y \subseteqq (Y^*_{fin})^{-1}X$. Supposons l'inclusion fausse, et prenons un $y \in Y - (Y^*_{fin})^{-1}X$. Posons

$$Z = \begin{cases} y^*(Y-y) & \text{si} \quad |y| < \omega, \\ Y-y & \text{si} \quad |y| = \omega. \end{cases}$$

Il est facile de vérifier $Z^+ = Y^*_{fin}(Y-y)$. D'où $X \subseteqq Z^* \subsetneqq Y^*$. Montrons que Z est un code. En effet, chaque élément α de Z^+ possède une factorisation unique en éléments de Y :

$$\alpha = y_1 \ldots y_n \quad \text{avec} \quad y_1 \ldots y_n \in Y^{(n)} \quad \text{et} \quad y_n \neq y.$$

Par conséquent, selon que $|y| < \omega$ ou $|y| = \omega$, α se présente uniquement sous la forme

$$\alpha = y^{p_1} y'_1 y^{p_2} y'_2 \ldots y^{p_r} y'_r \quad \text{avec} \quad y'_1 y'_2 \ldots y'_r \in (Y-y)^{(r)}, \quad p_i \geqq 0$$

$(i = 1, \ldots, r)$ où

$$\alpha = y_1 \ldots y_n \quad \text{avec} \quad y_1 \ldots y_n \in (Y-y)^{(n)},$$

c'est à dire α se factorise uniquement en éléments de Z. Donc Z^* est aussi un sous-monoïde quasi-libre contenant X, contrairement à la minimalité de $\overline{QL}(X) = Y^*$. L'inclusion $Y \subseteqq X(Y^*)^{-1}$ est démontrée d'une façon similaire en posant $Z = (Y-y)y^*$.

Théorème 3.2 (Théorème du défaut). *Soit* X *un langage infinitaire écrit par un code et soit* Y *la base de* $\overline{QL}(X)$. *Si* X *n'est pas un code, alors*

$$Card\,(Y) \leqq Card\,(X) - 1.$$

Preuve. Traitons tout d'abord le cas ou $\varepsilon \notin X$. Soit $\alpha : X \to Y$ l'application définie par :

$$\alpha(x) = y \quad \text{si} \quad x \in yY^*.$$

Elle est partout définie parce que $X \subseteqq Y^*$, et elle est univoque car Y est un code. Le Lemme 3.1 dit alors que α est surjective. X n'etant pas un code, il existe donc $x_1 \ldots x_n \in X^{(n)}$, $x'_1 \ldots x'_m \in X^{(m)}$ vérifiant

$$x_1 \ldots x_n = x'_1 \ldots x'_m, \quad x_1 \neq x'_1.$$

On en tire $\alpha(x_1) = \alpha(x'_1)$, donc α n'est pas injective, ce qui prouve l'inégalite annoncée.

Si $\varepsilon \in X$, on pose $X' = X - \varepsilon$. Clairement $\overline{QL}(X) = \overline{QL}(X')$. Si X' est un code, alors $X' = Y$ et on a $Card\ Y = Card\ X' = Card\ X - 1$; si X' n'est pas un code, alors par dessus $Card\ Y \leqq Card\ X' - 1 < Card\ X - 1$.

Corollaire 3.3. *Soit* $X = \{x_1, x_2\}$ *un langage infinitaire composé de deux mots. Alors* X *n'est pas un code ssi ou tous les deux mots de* X *sont puissances d'un même mot :*

$$x_1 = y^p, \quad x_2 = y^q \quad (p, q \geqq 0),$$

ou l'un déux est puissance ω *de l'autre :*

$$x_1 = x_2^{\omega} \quad \text{où} \quad x_2 = x_1^{\omega}.$$

Preuve ($\Leftarrow$) est évidente. Démontrons ($\Rightarrow$). Supposons que X ne soit pas un code. Trois cas sont possibles :

a) Au mois l'un des deux mots de X, disons x_1, est ε. Alors en prenant pour y le mot x_2 on a

$$x_1 = y^0, \quad x_2 = y.$$

b) x_1 et x_2 sont des mots finis non-vides. Alors X est écrit par un code, donc la base Y de $\overline{QL}(X)$, par le Théorème 3.2, se compose d'un seul mot, $Y=\{y\}$. Nous avons donc

$$x_1 = y^p, \quad x_2 = y^q$$

pour certains $p, q>0$.

c) L'un des deux mot de X, disons x_2, est un mot fini non-vide et l'autre est un mot infini. Alors $x_1=x_2^n x_1$ pour un $n>0$, d'où $x_1=x_2^\omega$.

4. Cas des langages infinitaires reconnaissable

Une partie X de A^∞ est *reconnaissable* s'il existe un morphisme $\varphi: A^\infty \to F$ de A^∞ sur un monoïde fini F qui sature X :

$$\varphi^{-1}\varphi(X) = X,$$

ou, d'une façon équivalente, s'il existe une congruence d'index fini θ de A^∞ qui sature X: X est union de classes de θ.

Le but de cette section est de montrer que si une partie infinitaire reconnaissable est écrite par un code, elle est aussi écrite par un code reconnaissable. Plus précisement nous établissons le résultat suivant qui est généralisation du Théorème 6.1 en [1] :

Théorème 4.1. *Soit X une partie de A^∞ qui est écrite par un code et soit Y la base de $\overline{QL}(X)$. Alors, si X est reconnaissable, Y l'est encore.*

La démonstration du Théorème 4.1 repose sur certaines propositions.

Proposition 4.2. *Si M est un sous-monoïde de A^∞ qui possède un ensemble générateur minimum distinct Y, et en particulier, si M est un sous-monoïde quasi-libre avec la base Y, alors M est reconnaissable ssi Y l'est encore.*

Preuve. Rappelons que la famille de parties reconnaissables de A^∞ est fermée par les opérations booléennes, par le produit et l'étoile (cf [2]). Donc M est reconnaissable si Y l'est. Pour démontrer la réciproque nous utilisons le Théorème 3 en [5] d'après lequel $Y=(M-\varepsilon)-(M_{fin}-\varepsilon)(M-\varepsilon)$. Il est facile de vérifier que pour toute X de A^∞, X_{fin} est reconnaissable si X l'est. Enfin $\{\varepsilon\}$ est clairement reconnaissable. Donc Y est reconnaissable si M l'est.

On associe maintenant à chaque partie Z de A^∞ deux parties U_Z et V_Z ainsi définies :

$$U_0 = V_0 = \{\varepsilon\}; \quad U_{j+1} = U_j^{-1}Z \cup Z^{-1}U_j; \quad V_{j+1} = ZV_j^{-1} \cup V_j Z^{-1} \quad j \geqq 0$$

$$U_Z = \bigcup_{j\geqq 0} U_j, \quad V_Z = \bigcup_{j\geqq 0} V_j.$$

Une congruence θ de A est *standard* si

$$\forall \alpha \in A^{\infty} \quad (\alpha \theta \varepsilon \rightarrow \alpha = \varepsilon).$$

À chaque congruence τ de A^{∞} on associe une congruence standard notée $\bar{\tau}$ définie par

$$\bar{\tau} = \tau \cap \mu,$$

où μ est la congruence dont les classes sont $\{\varepsilon\}$ et $A^{\infty} - \{\varepsilon\}$.

Proposition 4.3. *Soient Z une partie de A^{∞} et θ une congruence standard de A^{∞}. Si Z est saturée par θ, alors U_Z et V_Z sont également saturées par θ.*

Preuve. Montrons tout d'abord que pour toutes parties X et Y de A^{∞}, si X est saturée par θ, alors $Y^{-1}X$ et XY^{-1} le sont encore. En effet, soient $\alpha \theta \beta$ et $\alpha \in Y^{-1}X$. Montrons $\beta \in Y^{-1}X$. D'après la définition, il existe $\gamma \in Y$ tel que

$$\gamma\alpha \in X \quad \text{et} \quad |\gamma| = \omega \rightarrow \alpha = \varepsilon.$$

Comme X est saturée par θ et θ est standard, il en résulte

$$\gamma\beta \in X \quad \text{et} \quad |\gamma| = \omega \rightarrow \beta = \varepsilon,$$

ce qui signifie $\beta \in Y^{-1}X$. Donc $Y^{-1}X$ est saturé par θ. Pour XY^{-1} le raisonnement est similaire.

Évidemment U_0, V_0 sont saturés par θ. Par récurrence,

$$U_{j+1} = U_j^{-1}Z \cup Z^{-1}U_j \quad \text{et} \quad V_{j+1} = ZV_j^{-1} \cup V_j Z^{-1} \quad (j \geqq 0)$$

sont saturés par θ. Par conséquent

$$U_Z = \bigcup_{j \geqq 0} U_j \quad \text{et} \quad V_Z = \bigcup_{j \geqq 0} V_j$$

sont saturés par θ.

La proposition suivante dont la démonstration fait appel à plusieurs lemmes sera démontrée dans la section prochaine

Proposition. 4.4 *Si Z est une partie de A^{∞} qui contient le mot vide ε, alors*

$$[LB(Z^*)]^* = (U_Z \cap V_Z)^*.$$

Demonstration du Théorème 4.1 : Notons τ_X la congruence syntactique de X. Construisons une suite X_i de parties de A^{∞} comme suit :

$$\begin{cases} X_0 = X \cup \{\varepsilon\} \\ X_{i+1} = U_{X_i} \cap V_{X_i}, \quad i \geqq 0. \end{cases}$$

Alors, par la Proposition 4.4, $X_i^* = M_i^{LB}(X)$.

En vertu de la Proposition 4.3, chaque X_i est union de classes de la congruence standard $\bar{\tau}_X$. Si X est reconnaissable, alors $\bar{\tau}_X$ est d'index fini. Par conséquent le nombre des parties X_i différentes est fini. Donc il existe un n tel que $M^{LB}(X) = M_n^{LB}(X) = X_n^*$. D'où, par le Théorème 2.2, $Y^* = X_n^*$. En vertu de la Proposition 4.2, il en résulte que Y est reconnaissable.

5. Démonstration de la Proposition 4.4

Nous donnons d'abord quelques règles de calcul qui seront utilisées constamment dans la suite :

Lemme 5.1. *Soient R, S, T des parties de A^∞, et soit M un sous-monoïde de A^∞. Alors*

$$R \subseteq S \Rightarrow T^{-1}R \subseteq T^{-1}S, \quad R^{-1}T \subseteq S^{-1}T \tag{1}$$

$$R(ST)^{-1} = (RT^{-1})S^{-1} \tag{2}$$

$$(RS)^{-1}T = S^{-1}(R^{-1})T \quad \text{si} \quad \varepsilon \in S \tag{3}$$

$$(R^{-1}S)T^{-1} = R^{-1}(ST^{-1}) \quad \text{si} \quad \varepsilon \in T \tag{4}$$

$$(M^{-1}M)^{-1}M = M^{-1}M = (M^{-1}M)M \tag{5}$$

$$RM \cap SM^{-1} \subseteq (R \cap SM^{-1})M; \quad MR \cap M^{-1}S \subseteq M(R \cap M^{-1}S) \tag{6}$$

Preuve. Prouvons par exemple la formule (3). Soit $\alpha \in (RS)^{-1}T$. Ceci, par la définition, équivant à

$$\exists \beta \in R \exists \gamma \in S\colon\ (\beta\gamma)\alpha \in T \ \&\ (|\beta\gamma| = \omega \to \alpha = \varepsilon).$$

Deux cas sont possibles:

a) $|\beta| < \omega$. Alors la formule dernière implique

$$\exists \gamma \in S\colon\ (\exists \beta \in R\colon\ \beta(\gamma\alpha) \in T) \ \&\ (|\gamma| = \omega \to \alpha = \varepsilon)$$

ce qui équivant à

$$\exists \gamma \in R\colon\ (\gamma\alpha \in R^{-1}T) \ \&\ (|\gamma| = \omega \to \alpha = \varepsilon)$$

ce qui signifie que $\alpha \in S^{-1}(R^{-1}T)$.

b) $|\beta| = \omega$. Alors la formule indiquée plus haut implique $\alpha \in R^{-1}T$. Puisque $\varepsilon \in S$, ceci signifie que $\alpha \in S^{-1}(R^{-1}T)$.

Réciproquement, soit $\alpha \in S^{-1}(R^{-1}T)$. Il est facile de vérifier

$$\begin{aligned}
\alpha \in S^{-1}(R^{-1}T) &\Leftrightarrow \exists \beta \in S\colon\ (\beta\alpha \in R^{-1}T) \ \&\ (|\beta| = \omega \to \alpha = \varepsilon) \\
&\Leftrightarrow \exists \beta \in S\colon\ (\exists \gamma \in R\colon\ \gamma(\beta\alpha) \in T) \ \&\ (|\gamma| = \omega \to \beta\alpha = \varepsilon) \\
&\quad \&\ (|\beta| = \omega \to \alpha = \varepsilon) \\
&\Rightarrow \exists \beta \in S \exists \gamma \in R\colon\ (\gamma\beta)\alpha \in T \ \&\ (|\gamma\beta| = \omega \to \alpha = \varepsilon) \\
&\Leftrightarrow \alpha \in (RS)^{-1}T.
\end{aligned}$$

Posons, pour alléger l'écriture,

$$M = Z^*;\ \overline{U}_j = U_0 \cup U_1 \cup \ldots \cup U_j;\ \overline{V}_j = V_0 \cup V_1 \cup \ldots \cup V_j.$$

Lemme 5.2. *Pour tout $j \geqq 0$,*

$$U_{j+1} \subseteq (\overline{U}_j M)^{-1}Z \quad \text{et} \quad V_{j+1} \subseteq Z(M\overline{V}_j)^{-1}.$$

Preuve. Par récurrence sur j. Pour $j=0$, l'inclusion est vraie puisque $U_1 = Z \subseteq M^{-1}Z = (\overline{U}_0 M)^{-1}Z$. Si $j>0$, alors, par l'hypothèse d'induction et les formules

(1), (3), on a

$$U_{j+1} = U_j^{-1}Z \cup Z^{-1}U_j \subseteqq U_j^{-1}Z \cup Z^{-1}((\overline{U}_{j-1}M)^{-1}Z) =$$
$$= U_j^{-1}Z \cup (\overline{U}_{j-1}MZ)^{-1}Z = (U_j \cup \overline{U}_{j-1}MZ)^{-1}Z \subseteqq$$
$$\subseteqq (U_jM \cup \overline{U}_{j-1}M)^{-1}Z = (\overline{U}_jM)^{-1}Z,$$

ce qui prouve la première inclusion. La deuxième se démontre de la même façon en utilisant (2) au lieu de (3).

Lemme 5.3. *On a*

$$U_Z = M^{-1}U_Z, \quad V_Z = V_ZM^{-1},$$
$$M^{-1}M = U_ZM, \quad MM^{-1} = MV_Z.$$

Preuve. Par définition, $Z^{-1}U_j \subseteqq U_{j+1}$ pour $j \geqq 0$, donc $Z^{-1}U_Z \subseteqq U_Z$. De la même manière, $U_j^{-1}Z \subseteqq U_{j+1}$ $(j \geqq 0)$ implique

$$U_Z^{-1}Z \subseteqq U_Z. \tag{7}$$

De l'inclusion $Z^{-1}U_Z \subseteqq U_Z$, on obtient par récurrence sur n en utilisant (3) et (1).

$$(Z^{(n+1)})^{-1}U_Z = (Z^{(n)})^{-1}(Z_{fin}^{-1}U_Z) \subseteqq (Z^{(n)})^{-1}U_Z \subseteqq U_Z \quad n \geqq 0,$$

d'où, puisque $M = Z^* = \bigcup_{n \geqq 0} Z^{(n)}$, on a $M^{-1}U_Z \subseteqq U_Z^*$. L'inclusion $U_Z \subseteqq M^{-1}U_Z$ résulte de ce que $\varepsilon \in M$. La deuxième se démontre de la même façon.

Pour établir la troisième formule, nous vérifions par récurrence les inclusions

$$U_j \subseteqq M^{-1}M, \quad j \geqq 0.$$

Le cas $j=0$ étant evident. Par le Lemme 5.2 et par l'hypothèse d'induction, on obtient

$$U_{j+1} \subseteqq (\overline{U}_jM)^{-1}Z \subseteqq ((M^{-1}M)M)^{-1}Z.$$

Par (5) et (1)

$$U_{j+1} \subseteqq ((M^{-1}M)M)^{-1}Z = (M^{-1}M)^{-1}Z \subseteqq (M^{-1}M)^{-1}M = M^{-1}M.$$

D'où, $U_Z \subseteqq M^{-1}M$. Par conséquent

$$U_ZM \subseteqq (M^{-1}M)M = M^{-1}M.$$

Pour démontrer l'inclusion inverse $M^{-1}M \subseteqq U_ZM$ nous vérifions tout d'abord l'inclusion $(M^{-1}M)_{fin} \subseteqq U_ZM$. Supposons en effet $f \in (M^{-1}M)_{fin}$. Alors il existe $m \in M$ tel que

$$mf = m' \in M \quad \text{et} \quad (|m| = \omega) \to (f = \varepsilon).$$

Puisque $\varepsilon \in U_ZM$ on peut supposer $f \neq \varepsilon$ ce qui implique $|m| < \omega$.

Nous vérifions par récurrence sur $|mm'|$ que $f \in U_ZM$. Si $|mm'| = 0$, $f \in M \subseteqq U_ZM$; si $|mm'| \neq 0$, alors, puisque $M = Z^*$, il existe h, h' tels que

$$m = m_1h, \quad f = h'm_2 \quad \text{avec} \quad m_1, m_2 \in M, \quad hh' \in Z.$$

Alors $h \in (M^{-1}M)_{fin}$ avec $|mm_1| < |mm'|$. Par l'hypothèse d'induction, $h \in U_ZM$.

En utilisant (3) et (7) on obtient

$$h' \in (U_Z M)^{-1} Z = M^{-1}(U_Z^{-1} Z) \subseteq M^{-1} U_Z = U_Z.$$

Par conséquent $f = h' m \in U_Z M$.

Nous vérifions maintenant $(M^{-1}M)_{inf} \subseteq U_Z M$. Soit $u \in (M^{-1}M)_{inf}$. Il existe alors $m \in M_{fin}$ tel que $mu \in M$. D'une façon similaire au dessus, il existe h, h' tels que

$$m = m_1 h, \quad u = h' m_2 \quad \text{avec} \quad m_1, m_2 \in M, \quad hh' \in Z.$$

Alors $h \in (M^{-1}M)_{fin}$ ce qui implique $h \in U_Z M$. D'une façon similaire au dessus on a $h' \in U_Z$, par conséquent $u = h' m_2 \in U_Z M$.

Démonstration de la Proposition 4.4.

D'après le Lemme 5.3,

$$U_Z \cap V_Z \subseteq U_Z M \cap M V_Z = M^{-1}M \cap MM^{-1} = LB(M) = LB(Z^*).$$

Donc $(U_Z \cap V_Z)^* \subseteq [LB(Z^*)]^*$.

Réciproquement,

$$LB(M) = M^{-1}M \cap MM^{-1} = U_Z M \cap MM^{-1} \subseteq (U_Z \cap MM^{-1}) M.$$

Puis, par les formules deuxième et quatrième du Lemme 5.3 et par (6), on a

$$U_Z \cap MM^{-1} = U_Z \cap M V_Z = M^{-1} U_Z \cap M V_Z \subseteq M(M^{-1} U_Z \cap V_Z) = M(U_Z \cap V_Z).$$

D'où $LB(M) \subseteq M(U_Z \cap V_Z)M$. Comme $Z \subseteq U_Z \cap V_Z$, $M \subseteq (U_Z \cap V_Z)^*$, par $LB(M) \subseteq (U_Z \cap V_Z)^*$. Donc $[LB(Z^*)]^* \subseteq (U_Z \cap V_Z)^*$.

Abstract

We give necessary and sufficient conditions for an infinitary language to be written by an infinitary code. It is shown that the "Theorem of defect" remains valid for the case of infinitary languages and codes, and that if a recognizable infinitary language is written by an infinitary code, it can be also written by a recognizable infinitary code.

INSTITUTE OF MATHEMATICS
P/0 BOX 631 BO HO
HANOI, VIETNAM

References

[1] J. Berstel, D. Perrin, J. F. Perrot, A. Restivo, Sur le théorème du défaut, Journal of Algebra, 60 (1979), 169—180.

[2] S. Eilenberg, "Automata Languages and Machines", Vol. A, Academic Press, New York—London, 1974.

[3] Do Long Van, Codes avec des mots infinis, R.A.I.R.O. Informatique théorique, 16 (1982), 371—386.

[4] Do Long Van, Sous-monoïdes et codes avec des mots infinis, Semigroup Forum, 26 (1983), 75—87.

[5] Do Long Van, Sur les ensembles générateurs minimums des sous-monoïdes de A^{∞}, Preprint Series, Hanoi, 1984, №21.

[6] Do Long Van, Caractérisations combinatoires des sous-monoïdes engendrés par un code infinitaire, Preprint Series, Hanoi—1984, № 6.

(Received Febr. 3, 1985)

On the congruences of finite autonomous Moore automata

By A. ÁDÁM

1. Introduction

By a congruence of an automaton, a partition π of the set of its states is meant such that π is compatible both with the transition function and with the output function. The general problem of describing the congruences of finite Moore automata seems to be a very difficult question.

In the present paper, the congruences of (possibly non-connected) finite Moore automata which have *only one* input sign are presented by a recursive construction. After introducing the most important notions, the question is elucidated in three phases. (The first and third phases are almost trivial.) First, an overview of the congruences of cyclic automata[1] is given in Section 3. The second phase is the single stage of the procedure which requires labour; in this phase the congruences possessing the following property are obtained by a construction: whenever a is a cyclic state, then the congruence class containing a intersects every connected component of the automaton (Section 4). This result can easily be extended into a complete solution of the main problem of the paper (Section 5).

The considerations of Section 4 are illustrated by an example in Section 6.

The final section of the paper gives a broad survey of several problems concerning the congruences of finite Moore automata; some related earlier investigations are referred to here, too. If the reader wants first to get a comprehensive overview of a variety of problems, and thereafter to narrow down his interest to the particular question analyzed actually, then he can be recommended to begin the study of the paper with Section 7.

The author wishes to express his gratitude to the referee, Dr. GY. POLLÁK, for his various suggestions which made the considerations clearer at several places of the paper, primarily in section 4.

[1] The attribute "cyclic" is used in the sense that the graph of the automaton is a (directed) cycle. (In some articles, the same attribute is used to mean that the automaton has a state which constitutes a one-element generating system.)

2. Terminology

We shall use the standard terminology of automaton theory and certain basic notions in graph theory without explicit definitions.[2] We shall consider automata so that no state is distinguished in them as an initial one, and (if not otherwise stated) we do not pose any connectivity restriction.

A finite Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ is called *autonomous* if the input set X consists of a single element x. The automata, studied in this paper, are thought to be autonomous (unless otherwise stated). The graph-theoretical structure of these automata is described by the (simple but important) well-known theorem of Ore ([8], § 4.4; [1], Chapter I). Denote the connected components of $\mathbf{A}$ by $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_t$. Ore's theorem implies that

(i) each connected component $\mathbf{A}_i$ (where $1 \leqq i \leqq t$) contains exactly one cycle $\mathbf{Z}_i$,

(ii) $\mathbf{A}_i$ has no other circuit than $\mathbf{Z}_i$,

(iii) an edge of $\mathbf{A}_i$ which does not belong to $\mathbf{Z}_i$ is directed towards $\mathbf{Z}_i$.

A state a is called *cyclic* if a belongs to the cycle of the connected component containing a. In the contrary case, a is called an *acyclic* state.

Let a, b be two states of an automaton. Define $\chi(a, b)$ as the smallest non-negative number i such that $\delta(a, x^i)=b$. (Possibly $\chi(a, b)$ is undefined.)

Connected components and cycles are, obviously subautomata of $\mathbf{A}$. Let a be a state; we denote by $\mathbf{A}[a]$ the connected component containing a and by $\mathbf{Z}[a]$ the cycle of $\mathbf{A}[a]$.

The next evident assertion yields a recursive description of the subautomata of $\mathbf{A}$.

Proposition 1. *Let* $\mathbf{A}$ *be an (autonomous) automaton. Then*

(i) *the union of an arbitrary number* $(\geqq 1)$ *of cycles is a subautomaton of* $\mathbf{A}$,

(ii) *whenever* $\mathbf{B}=(B, \{x\}, Y, \delta, \lambda)$ *is a subautomaton and* a *is a state of* $\mathbf{A}$ *such that*

$$a \notin B \ \& \ \delta(a, x) \in B,$$

then $\mathbf{C}=(B \cup \{a\}, \{x\}, Y, \delta, \lambda)$ *is a subautomaton,*

(iii) *each subautomaton of* $\mathbf{A}$ *can be obtained by applying* (i), (ii) (*where* (ii) *is applied several — possibly zero — times*).

Let a be an arbitrary state of $\mathbf{A}$. The smallest i such that $\delta(a, x^i)$ belongs to $\mathbf{Z}[a]$ is called the *height* of a. We denote by M_i the set of all states of height i. (Hence M_0 is the set of cyclic states; $M_0 \cup M_1 \cup \ldots \cup M_j$ constitutes a subautomaton for each j $(\geqq 0)$.)

A partition π of the state set A of an (autonomous) automaton $\mathbf{A}$ is called a *congruence* (of $\mathbf{A}$) if $a \equiv b \pmod{\pi}$ implies

$$\delta(a, x) \equiv \delta(b, x) \pmod{\pi}$$

and

$$\lambda(a) = \lambda(b).$$

[2] In particular, "cycle" is understood as a directed graph and the word "circuit" is used if we do not take orientation into account.

For each congruence π, we can introduce the *factor automaton* $\mathbf{A}/\pi$ so that A/π is the state set of $\mathbf{A}/\pi$ and the functions δ, λ are defined in $\mathbf{A}/\pi$ in the natural manner.

The minimal partition o of A is always a congruence. The automaton $\mathbf{A}$ is called *simple* (or *reduced*) if $\mathbf{A}$ has no other congruence than the minimal partition of A. It is easy to see that, for an arbitrary automaton $\mathbf{A}$, there exists a maximal congruence[3] π_{max}, moreover, $\mathbf{A}/\pi$ is simple precisely in the case $\pi=\pi_{max}$.

An isomorphism between automata is understood as a state-isomorphism, an analogous agreement holds for homomorphisms.

Let us define a partition π_c of A such that two states a, b are in a common class modulo π exactly if they are in the same connected component. π_c fails to be a congruence in general.

A partition π of the state set of an automaton $\mathbf{A}$ is called *extensive* if each class modulo π which contains at least one cyclic state meets every connected component. (In other words, more explicitly: π is said extensive if, whenever to a pair a, b of states there exists a positive number j satisfying $\delta(a, x^j)=a$, then there is a state c which fulfils $a\equiv c \pmod{\pi}$ and $b\equiv c \pmod{\pi_c}$.)

Consider two connected components $\mathbf{A}_i$, $\mathbf{A}_j$ of $\mathbf{A}$. Denote the maximal congruences of the cycles $\mathbf{Z}_i$, $\mathbf{Z}_j$ by π_i and π_i, respectively. If Z_i/π_j and $\mathbf{Z}_j/\pi_j$ are isomorphic automata, then we call $\mathbf{A}_i$ and $\mathbf{A}_j$ *similar* components. The similarity is an equivalence relation in the set of all connected components of the automaton. An automaton $\mathbf{A}$ is called *pan-similar* if every pair of connected components of $\mathbf{A}$ is similar. (A connected automaton is trivially pan-similar.)

3. The congruences of cyclic automata

Consider an automaton $\mathbf{A}$ such that $\mathbf{A}$ is a cycle. (See Fig. 1.) Denote the number of states (i.e., the length of the cycle) by v. Suppose that the states of $\mathbf{A}$ are denoted by $a_1, a_2, \ldots, a_v$ so that

$$\delta(a_1, x) = a_2,\ \delta(a_2, x) = a_3, \ldots, \delta(a_{v-1}, x) = a_v,\quad \delta(a_v, x) = a_1.$$

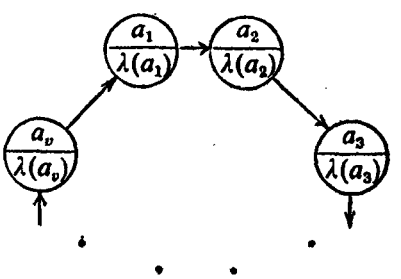


Fig. 1.

[3] The maximality means that each congruence π is a refinement of π_{max}. In general, π_{max} is not equal to the maximal partition ι of A.

Let s be the smallest number[4] such that the v equalities

$$\lambda(a_1) = \lambda(a_{1+s}), \quad \lambda(a_2) = \lambda(a_{2+s}), \ldots, \lambda(a_{v-s}) = \lambda(a_v),$$

$$\lambda(a_{v-s+1}) = \lambda(a_1), \quad \lambda(a_{v-s+2}) = \lambda(a_2), \ldots, \lambda(a_v) = \lambda(a_s) \tag{3.1}$$

are true. s is called the *periodicity number* of **A**. We have clearly $1 \leqq s \leqq v$. The cycle is called *primitive* or *imprimitive* according as $s=v$ or $s<v$ holds.

It is obvious that the periodicity number s is a divisor of the cycle length v.

Construction I. Choose an integer d such that $s|d|v$. Introduce the partition π_d of A by

$$a_i \equiv a_j \pmod{\pi_d} \Leftrightarrow d|j-i$$

(where $1 \leqq i \leqq v$, $1 \leqq j \leqq v$).

The index of π_d is d. Each class modulo π_d has v/d elements.

Theorem 1. *A partition π of the state set A of a cyclic automaton* **A** *is a congruence of A if and only if there exists a number d such that* ($s|d|v$ *and*) $\pi=\pi_d$.

Proof. Sufficiency is evident. — Consider an arbitrary congruence π of **A**. If we define d as the smallest positive number such that $a \equiv b \pmod{\pi}$ for suitable states satisfying $\chi(a, b)=d$, then it is easy to see that $\pi=\pi_d$.

Corollary 1. *The congruence lattice of* **A** *is isomorphic to the lattice of divisors of v/s.*

Proof. Let d^* be an arbitrary divisor of v/s, let us assign to d^* the congruence π_{v/d^*}. It is easy to see that this assignment is an isomorphism.

The following assertions are immediate consequences of our former considerations:

Corollary 2. *The maximal congruence of* **A** *is π_s. Among the factor automata* $\mathbf{A}/\pi_d$ *(where d runs through the numbers fulfilling $s|d|v$) only* $\mathbf{A}/\pi_s$ *is reduced.* **A** *is reduced if and only if* **A** *is a primitive cycle.*

4. The extensive congruences of pan-similar automata

4.1. Introductory considerations

Let **A** be a pan-similar automaton. Consider an arbitrary state a of **A**, let i be the height of a. There is a state b, determined by a uniquely, such that b belongs to $\mathbf{Z}[a]$ and

$$\delta(b, x^i) = \delta(a, x^i).$$

We shall denote b by $\sigma(a)$. Thus we have defined an idempotent mapping σ of the set of all states onto the set of cyclic states. It can be seen easily that $\sigma(\delta(a, x)) = \delta(\sigma(a), x)$.

[4] The existence of s follows from the fact that the formulae (3.1) are valid for v (instead of s).

Denote by $\mathbf{D}=(D, \{x\}, Y, \delta, \lambda)$ the largest subautomaton of **A** which satisfies the implication

$$a\in D \Rightarrow \lambda(a) = \lambda(\sigma(a)).$$

The following statements are obvious.

Lemma 1.

(I) **D** *exists and includes all the cycles of* **A**.

(II) *D can be obtained also as the smallest subset of A fulfilling the following two requirements:*

(A) *Every cyclic state belongs to D.*

(B) *If a is acyclic,* $\delta(a, x)\in D$ *and* $\lambda(a)=\lambda(\sigma(a))$, *then* $a\in D$.

(III) *The formulae* $a\in D$ *and* $a\equiv\sigma(a) \pmod{\pi_{\max}}$ *are equivalent (where* $a\in A$ *and* $\pi_{\max}$ *is the maximal congruence of* **A**).

Since we have supposed that **A** is a pan-similar automaton, there exists a cyclic automaton **Z** such that **Z** is isomorphic to each $\mathbf{Z}_k/\pi_k$ where π_k is the maximal congruence of the cycle $\mathbf{Z}_k$ of the connected component $\mathbf{A}_k$ of **A**. (k runs from 1 to t, where t is the number of components.) **Z** is primitive. For each choice of k, there is exactly one homomorphism τ_k from $\mathbf{Z}_k$ onto **Z**.

Denote the number of states of **Z** by s and, for any choice of k, the number of states of $\mathbf{Z}_k$ by v_k. (Clearly $s|v_k$.)

Lemma 2. *Let* a, b *be two elements of D. Define* k *and* m *by* $\mathbf{Z}_k=\mathbf{Z}[a]$, $\mathbf{Z}_m=\mathbf{Z}[b]$. *If* $\tau_m(\sigma(a))=\tau_k(\sigma(b))$, *then* $\lambda(a)=\lambda(b)$.

Proof. We have

$$\lambda(a) = \lambda(\sigma(a)) = \lambda^*(\tau_k(\sigma(a))) = \lambda^*(\tau_m(\sigma(b))) = \lambda(\sigma(b)) = \lambda(b),$$

where λ^* is the output function of **Z**. Indeed, the first and fifth equalities are valid by the definition of **D**, the second and fourth ones hold because τ_k, τ_m are homomorphisms.

4.2. Recursive description of the extensive congruences

Construction II.

Step 1. Choose a subautomaton $\mathbf{G}_0=(G_0, \{x\}, Y, \delta, \lambda)$ of **A** such that $\mathbf{G}_0$ is included in **D** and each cycle $\mathbf{Z}_k$ is included in $\mathbf{G}_0$.

Step 2. Define an ascending sequence

$$\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \ldots$$

of subautomata of **A** so that[5] $a\in G_{i+1}$ if and only if $\delta(a, x)\in G_i$. (The sequence is finished when **A** is entirely exhausted.)

Step 3. Choose a number d such that $s|d$ and d is a common divisor of the cycle lengths $v_1, v_2, \ldots, v_t$. Choose, furthermore, a sequence $z_1, z_2, \ldots, z_t$ of

[5] Of course, G_i is here the set of states of $\mathbf{G}_i$.

states such that z_k belongs to the cycle $\mathbf{Z}_k$ $(1 \leqq k \leqq t)$ and the equalities

$$\tau_1(z_1) = \tau_2(z_2) = \ldots = \tau_t(z_t)$$

hold.

Step 4. Introduce a sequence of partitions $\pi^{(0)}, \pi^{(1)}, \pi^{(2)}, \ldots$ in the following (recursive) manner:

(I) Each $\pi^{(i)}$ is a partition of G_i.

(II) Two elements a, b of G_0 are congruent modulo $\pi^{(0)}$ exactly if

$$\chi(a, z_k) \equiv \chi(b, z_m) \pmod{d},$$

where k and m are defined by $\mathbf{Z}_k = \mathbf{Z}[a]$ and $\mathbf{Z}_m = \mathbf{Z}[b]$.

(III) Suppose that $\pi^{(i)}$ has already been defined. Introduce $\pi^{(i+1)}$ so that the following three rules be observed:

(α) If $a \in G_i$ and $b \in G_i$, then $a \equiv b \pmod{\pi^{(i+1)}}$ holds precisely when $a \equiv b \pmod{\pi^{(i)}}$.

(β) If $a \in G_i$ and $b \in G_{i+1} - G_i$, then $a \not\equiv b \pmod{\pi^{(i+1)}}$.

(γ) If a and b belong to $G_{i+1} - G_i$ and $a \equiv b \pmod{\pi^{(i+1)}}$, then $\lambda(a) = \lambda(b)$ and $\delta(a, x) \equiv \delta(b, x) \pmod{\pi^{(i)}}$.

(It is clear that (γ) admits a certain liberty in partitioning the elements of $G_{i+1} - G_i$ into classes.)

Step 5. Denote by π the partition $\pi^{(i^*)}$ with the largest possible superscript i^*. (Obviously, π is a partition of $G_{i^*} = A$.)

Lemma 3. *If* $a \equiv b \pmod{\pi}$, *then* $\lambda(a) = \lambda(b)$.

Proof. Suppose $a \equiv b \pmod{\pi}$. There exists a subscript i such that a, b belong to G_i but (if $i > 0$) they are not contained in G_{i-1}. The proof proceeds by induction on i.

Let a, b be elements of $G_0 (\subseteq D)$, recall (II) in Step 4 of Construction II. We have

$$\chi(\sigma(a), z_k) \equiv \chi(a, z_k) \equiv \chi(b, z_m) \equiv \chi(\sigma(b), z_m) \pmod{d}$$

(the first and third congruences are clearly true modulo v_k, v_m, resp., this implies their validity modulo d), hence $\tau_k(\sigma(a)) = \tau_m(\sigma(b))$, thus $\lambda(a) = \lambda(b)$ by Lemma 2.

Assume that the lemma is valid for i. Let a, b be elements of $G_{i+1} - G_i$ such that they are congruent modulo π. Then they are congruent also modulo $\pi^{(i+1)}$. $\lambda(a) = \lambda(b)$ follows from the rule (γ) in the item (III) of Step 4 of Construction II.

Lemma 4. π *is a congruence.*

Proof. After the preceding lemma, it suffices to show that $a \equiv b \pmod{\pi}$ implies $\delta(a, x) \equiv \delta(b, x) \pmod{\pi}$.

Let $a \equiv b \pmod{\pi}$ hold. There is an i as in the previous proof. Again, we use induction. First we consider the case $i = 0$. Use the short notations $a' = \delta(a, x)$ and $b' = \delta(b, x)$, recall item (II) of Step 4 of Construction II. We have

$$\chi(a', z_k) \equiv \chi(a, z_k) - 1 \equiv \chi(b, z_m) - 1 \equiv \chi(b', z_m) \pmod{d},$$

where the second congruence follows from $a \equiv b \pmod{\pi}$, the first and third congru-

ences are valid[6] because d is a divisor of the lengths of the cycles containing z_k and z_m. Hence $a' \equiv b' \pmod{\pi}$.

If i is positive, then the inference

$$a \equiv b \pmod{\pi} \Rightarrow a \equiv b \pmod{\pi^{(i)}} \Rightarrow \delta(a, x) \equiv \delta(b, x) \pmod{\pi^{(i-1)}} \Rightarrow$$

$$\Rightarrow \delta(a, x) \equiv \delta(b, x) \pmod{\pi}$$

is valid according to item (III) of Step 4 of the construction.

Theorem 2. *A partition π of A is an extensive congruence of* **A** *if and only if π can be obtained by Construction II.*

Proof.

Sufficiency. Having Lemma 4, we are going to show the extensivity of a congruence π obtained by the construction. Assume that a belongs to $\mathbf{Z}_k$ and b belongs to $\mathbf{A}_m$, we want to find a $c(\in A_m)$ with $a \equiv c \pmod{\pi}$. The choice $c = \delta(z_m, x^\chi)$ is convenient, where χ stands shortly for $\chi(z_k, a)$.

Necessity. Let an extensive congruence π of **A** be considered. Our next aim is to determine the circumstances (more precisely: the choices of $d, z_1, z_2, \ldots, z_t, G_0, \pi^{(0)}, G_1, \pi^{(1)}, G_2, \pi^{(2)}, \ldots$) under which just the prescribed π is obtained by Construction II.

Let G_0 be the set of states $a(\in A)$ for which there is a cyclic state c such that $a \equiv c \pmod{\pi}$. Let G_{i+1} (where i can be $0, 1, 2, \ldots$) be the set of states a satisfying $\delta(a, x) \in G_i$. Let $\pi^{(i)}$ be the restriction of π to the set G_i.

Let $z_1, z_2, \ldots, z_t$ be arbitrary states in the cycles $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_t$, respectively, such that they are pairwise congruent modulo π.

Choose a cyclic state z and denote by d the smallest positive number which satisfies $z \equiv \delta(z, x^d) \pmod{\pi}$. It can be seen that d does not depend on the choice of z.

Let π^* be the congruence which is yielded by Construction II with the parameters introduced above and with a suitable application of (III/γ) in Step 4. We want to show $\pi^* = \pi$. Consider two states a, b; we are going to get that they are congruent modulo π^* exactly when they are congruent modulo π.

Suppose first $a \in G_0$ and $b \in G_0$. Consider the three statements

$$a \equiv b \pmod{\pi^*},$$

$$\chi(a, z_a) \equiv \chi(b, z_b) \pmod{d},$$

$$a \equiv b \pmod{\pi}.$$

It can be seen that the second statement is equivalent both to the first and the third one.

We turn to the case $a \in G_i$, $b \in G_{i+1} - G_i$. With this choice of a and b, we have $a \not\equiv b \pmod{\pi^*}$. On the other hand, $a \equiv b \pmod{\pi}$ would imply

$$\delta(b, x^{i+d}) \equiv \delta(a, x^{i+d}) \equiv \delta(a, x^i) \equiv \delta(b, x^i) \pmod{\pi}$$

[6] Except the possibility $a = z_k$, the equality $\chi(a', z_k) = \chi(a, z_k) - 1$ is also true (and analogously for b).

(the second congruence follows from $\delta(a, x^i) \in G_0$), and this is impossible since

$$\delta(b, x^i) \in G_1 - G_0.$$

By getting a contradiction, $a \not\equiv b \pmod{\pi}$ is verified.

Finally, assume that a and b belong to the same $G_{i+1} - G_i$. The equivalence of $a \equiv b \pmod{\pi}$ and $a \equiv b \pmod{\pi^*}$ follows from the fact that we have defined $\pi^{(i+1)}$ as the restriction of π. It remained still dubious whether or not the sequence

$$\pi^{(0)}, \pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(i^*)}$$

(as we have derived it from π) satisfies (III/γ) in Step 4 of Construction II. This holds, however, because π is a congruence.

By analyzing Construction II and Theorem 2, we get the following result:

Corollary 3. *The maximal congruence $\pi_{\max}$ of* **A** *is extensive, and just $\pi_{\max}$ is obtained when we apply Construction II in the following manner: d is chosen as equal to s; $\mathbf{G}_0$ is chosen as equal to* **D**; *for each possible value of i, let $a \equiv b \pmod{\pi^{(i+1)}}$ hold precisely when both $\lambda(a) = \lambda(b)$ and*

$$\delta(a, x) \equiv \delta(b, x) \pmod{\pi^{(i)}}$$

are true (where a and b belong to $G_{i+1} - G_i$).

4.3. The question of unicity

Construction I has yielded uniquely the congruences of cycles. (Also Constructions III, IV will prove to be unique.) It may happen, however, that two *different* applications of Construction II lead to the *same* extensive congruence. More nearly: if we modify either G_0 or d or the $\pi^{(i)}$'s, then the obtained congruence π is necessarily altered; but it is possible that two different systems of form $z_1, z_2, \ldots, z_t$ give the same congruence.

Proposition 2. *Let two realizations of Construction II be considered. Suppose that $d, G_0, \pi^{(0)}, G_1, \pi^{(1)}, G_2, \pi^{(2)}, \ldots$ are common in them. Denote the states which represent the cycles by $z_1, z_2, \ldots, z_t$ in the first execution, and by $z_1', z_2', \ldots, z_t'$ in the second one. Denote the obtained congruences by π and π', respectively. Then $\pi = \pi'$ if and only if the numbers*

$$\chi(z_1, z_1'), \chi(z_2, z_2'), \ldots, \chi(z_t, z_t')$$

are congruent to each other modulo d.

Next we show two lemmas.

Lemma 5. *First apply Construction II with the system $z_1, z_2, \ldots, z_t$, and then modify the application in such a way that the system of the z_i's is replaced by the system*

$$z_1^* = \delta(z_1, x), \quad z_2^* = \delta(z_2, x), \ldots, z_t^* = \delta(z_t, x).$$

Both realizations of Construction II give the same congruence.

Proof. The statement is implied by the construction (especially, item (II) of Step 4) and the deduction

$$\chi(a, z_k^*) \equiv \chi(a, z_k)+1 \equiv \chi(b, z_m)+1 \equiv \chi(b, z_m^*) \pmod{d}.$$

Lemma 6. *Apply Construction II with the system* $z_1, z_2, \ldots, z_t$, *select a number* i $(1 \leqq i \leqq t)$ *and modify the application in such a way that* z_i *is replaced by* $z_i^+ = \delta(z_i, x^d)$. *Both realizations give the same congruence.*

Proof. It is easy to see that

$$\chi(a, z_i^+) \equiv \chi(a, z_i) \pmod{d}$$

for each state a of $\mathbf{A}_i$; hence the statement follows immediately.

Proof of Proposition 2.

Sufficiency. Consider the system $z_1, z_2, \ldots, z_t$. First apply Lemma 5 $\chi(z_1, z_1')$ times, thus we get a system $z_1^*, z_2^*, \ldots, z_t^*$ such that $z_1^* = z_1'$ and $d \mid \chi(z_i^*, z_i')$ for each i $(2 \leqq i \leqq t)$. We can obtain the system $z_1', z_2', \ldots, z_t'$ by applying Lemma 6 (several times, in a straightforward manner).

Necessity. Suppose

$$\chi(z_i, z_i') \not\equiv \chi(z_j, z_j') \pmod{d}$$

for a suitable pair i, j $(1 \leqq i \leqq t,\ 1 \leqq j \leqq t)$. Then z_i and z_j are congruent modulo π, and it is easy to see that they are incongruent modulo π'. Hence $\pi \neq \pi'$.

4.4. Considerations on how certain subautomata can be generated

Construction II relies upon the subautomata of **D** containing all the cyclic states. From a theoretical point of view, Proposition 1 gives a good survey of these subautomata.

This survey has the practical disadvantage that a subautomaton is handled as the set of *all* states of it. It would be more useful, to characterize the subautomata in terms of certain sets which consist of a relatively *small* number of states. The present subsection is devoted to this subject.

Let $\mathbf{B}=(B, \{x\}, Y, \delta, \lambda)$ be a subautomaton of **A** such that **B** includes each cycle. Denote by $R(\mathbf{B})$ the set of states a satisfying the condition

$$a \in B \ \& \ (\forall b)[b \in B \Rightarrow \delta(b, x) \neq a].$$

$R(\mathbf{B})$ is called the *minimal generating system* of **B**. Each element of $R(\mathbf{B})$ is an acyclic state. (If, in particular, **B** is the union of all cycles, then $R(\mathbf{B})=\emptyset$.)

It is evident that a state b belongs to B if and only if either b is cyclic or there is an $a(\in R(\mathbf{B}))$ and a number $i(\geqq 0)$ such that $\delta(a, x^i)=b$.

Proposition 3. *If* $\mathbf{B}_1$ *and* $\mathbf{B}_2$ *are different subautomata of* **A** *which contain all the cyclic states, then* $R(\mathbf{B}_1) \neq R(\mathbf{B}_2)$.

Proof. If $R(\mathbf{B}_1)=R(\mathbf{B}_2)$, then $\mathbf{B}_1$ equals $\mathbf{B}_2$ in consequence of the sentence before the proposition.

Proposition 4. *Let R be a (possibly empty) set of acyclic states. The following statements* (A), (B) *are equivalent:*

(A) *There exists a subautomaton* **B** *of* **A** *such that* **B** *contains all the cyclic states,* **B** *is a subautomaton of* **D** *and* $R(\mathbf{B})=R$.

(B) *R is a subset of D and whenever $a\in R$ and i is a positive number, then $\delta(a, x^i)\notin R$.*

Proof. (A)$\Rightarrow$(B) is evident. — If a set R satisfies (B), then it is easy to see that a is acyclic and $R=R(\mathbf{B})$ holds for the subautomaton **B** which is defined by the following rule: $b\in B$ if and only if either b is cyclic or there is an $a(\in R)$ and a non-negative number i such that $\delta(a, x^i)=b$.

Construction III. The construction consists of an initial step and an arbitrary number ($\geqq 0$) of general steps.

Initial step. Let R_1 be an arbitrary non-empty subset of $M_1\cap D$.

General step. Consider a set R_i such that R_i has been obtained by the preceding step of the construction, $R_i\subseteqq M_1\cup M_2\cup\ldots\cup M_i$ and $R_i\cap M_i\neq\emptyset$. Choose a non-empty subset Q of $R_i\cap M_i$ such that $\delta^{-1}(q)\cap D\neq\emptyset$ for each choice of $q\in Q$, where $\delta^{-1}(q)$ is the set of states a satisfying $\delta(a, x)=q$. Choose for each $q(\in Q)$ a non-empty subset $\theta(q)$ of $\delta^{-1}(q)\cap D$. Let us form the set

$$R_{i+1}=(R_i-Q)\cup\Big(\bigcup_{q\in Q}\theta(q)\Big).$$

Construction III can be finished after an arbitrary step. It breaks up necessarily when there is no possibility for the non-empty choice of Q.

Proposition 5. *The realizations of Construction III give pairwise different sets. A set R is obtainable by Construction III if and only if $R=R(\mathbf{B})$ with some subautomaton* **B** *such that* **B** *contains all the cyclic states,* **B** *is included in* **D**, *and* **B** *has at least one acyclic state.*

Proof. The first assertion follows from the requirements that certain sets must be non-empty in Construction III. The second assertion is an easy consequence of the characterization of the sets $R(\mathbf{B})$ stated in Proposition 4.

5. Overview of the congruences in the general case

Let **A** be an arbitrary finite autonomous Moore automaton. Denote by π_h the partition of A such that $a\equiv b\pmod{\pi_h}$ holds precisely if the connected components which contain a and b are similar. Evidently, $\pi_c\subseteqq\pi_h$.

Construction IV.

Step 1. Let a partition π^* of A be chosen such that $\pi_c\subseteqq\pi^*\subseteqq\pi_h$. Denote by $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_q$ the (pan-similar) subautomata of **A** which are determined by the classes $A_1, A_2, \ldots, A_q$ modulo π^*, respectively. (q is the index of π^*.)

Step 2. For each choice of i ($1\leqq i\leqq q$), let us consider a partition π_i of A which satisfies the following assertions:

(i) $A_1 \cup A_2 \cup \ldots \cup A_{i-1} \cup A_{i+1} \cup \ldots \cup A_q$ is (precisely) one class modulo π_i.
(ii) The restriction of π_i to A_i is an extensive congruence of $\mathbf{A}_i$.

Step 3. Let us form the partition

$$\pi = \pi_1 \cap \pi_2 \cap \ldots \cap \pi_q$$

of A.

Theorem 3. *A partition π of A is a congruence of* **A** *if and only if π can be obtained by Construction IV.*

Proof. Sufficiency is evident. — Consider a congruence π of **A**. If we take π^* as $\pi \cup \pi_c$ and define each π_i so that π_i coincides with π on A_i, then it is clear that Construction IV gives π.

An easy consequence of the previous considerations of Section 5 is:

Corollary 4. *The maximal congruence of* **A** *is obtained when we choose (in Construction IV) π^* as equal to π_h and we determine each π_i so that its restriction to A_i should be the maximal congruence of $\mathbf{A}_i$.*

Proposition 6. *An automaton* **A** *is reduced if and only if the following three assertions hold:*
(i) *Each cycle of* **A** *is primitive.*
(ii) *The cycles of* **A** *are pairwise non-isomorphic.*
(iii) *There is no pair of different states a, b in* **A** *such that $\delta(a, x)=\delta(b, x)$ and $\lambda(a)=\lambda(b)$.*

Proof.

Necessity. If (i) does not hold, then we get a nontrivial congruence so that we select an imprimitive cycle **Z** and we define π so that $a \equiv b \pmod{\pi}$ if either $a=b$ or a, b are states of **Z** which satisfy $s|\chi(a, b)$.

If (ii) is not true, then we can choose two different cycles and an isomorphism α between them; the following partition π is a nontrivial congruence: $a \equiv b \pmod{\pi}$ is either $a=b$ or one of a, b is the image of the other under α.

If (iii) is not valid, then let us choose a pair a, b fulfilling $\lambda(a)=\lambda(b)$ and $\delta(a, x)=\delta(b, x)$; the following partition is a nontrivial congruence: $\{a, b\}$ is one of the classes and all other classes consist of one element.

Sufficiency. Suppose that (i), (ii), (iii) are fulfilled. It is clear that $\pi_c=\pi_h$. Let us recall the considerations of Section 4 in case of an arbitrary connected component $\mathbf{A}_i$ of **A**. D consists of the cyclic states only. Corollary 3 and the last sentence of Corollary 2 imply that the maximal congruence of $\mathbf{A}_i$ equals its minimal congruence, i.e., $\mathbf{A}_i$ is simple. Taking Corollary 4 into account, we get that also **A** is reduced.

Remark 1. Consider the conditions (i), (ii) in Proposition 6. (i) & (ii) can be formulated in the following manner (equivalently):
(iv) Whenever $\mathbf{Z}_1, \mathbf{Z}_2$ are cyclic subautomata of **A** and there is an isomorphism α of $\mathbf{Z}_1$ onto $\mathbf{Z}_2$, then ($\mathbf{Z}_1=\mathbf{Z}_2$ and) α is the identical automorphism of $\mathbf{Z}_1$.

Remark 2. The sufficiency of the conditions in Proposition 6 can be proved also by using the following idea (without any reference to the previous results):

we start with a congruence π and two different states such that $a \equiv b \pmod{\pi}$, and we strive to show by studying the sequences

$$a, \delta(a, x), \delta(a, x^2), \ldots$$

and

$$b, \delta(b, x), \delta(b, x^2), \ldots$$

that either (i) or (ii) or (iii) is violated.

The question may arise when two congruences, obtained either by Construction II or by Construction IV, are related in such a way that one is a refinement of the other. The answer is given in the next results which can be verified by routine inferences.

Proposition 7. *Consider two realizations of Construction II (concerning the same automaton* **A***). Distinguish them from each other by the sub- or superscripts* α *and* β*; in particular, let the obtained congruences be* π_α *and* π_β*, respectively. The relation* $\pi_\alpha \leqq \pi_\beta$ *holds if and only if the following four conditions are satisfied:*

(A) $G_0^\alpha \leqq G_0^\beta$.

(B) $d_\beta \mid d_\alpha$.

(C) *The numbers*

$$\chi(z_1^\alpha, z_1^\beta),\ \chi(z_2^\alpha, z_2^\beta), \ldots, \chi(z_t^\alpha, z_t^\beta)$$

are congruent to each other modulo d_β.

(D) *Whenever two different states* a *and* b *are congruent* mod π_α *in consequence of* (III/γ) *in Step 4 of the (first execution of) Construction II and they are not contained in* G_0^β*, then* a *and* b *belong to the same* $\mathbf{G}_{j+1}^\beta$ *and they are in a common class* mod $\pi_\beta^{(j+1)}$ *(in course of Step 4 of the second realization).*

Proposition 8. *Consider two realizations of Construction IV (concerning the same automaton* **A***). Distinguish them from each other as in the preceding proposition. The relation* $\pi_\alpha \leqq \pi_\beta$ *holds if and only if the following conditions* (I), (II) *are fulfilled:*

(I) $\pi_\alpha^* \leqq \pi_\beta^*$.

(II) *The implication*

$$a \equiv b \pmod{\pi_i^\alpha} \Rightarrow a \equiv b \pmod{\pi_j^\beta}$$

is valid for every i $(1 \leqq i \leqq q_\alpha)$*, where* j $(1 \leqq j \leqq q_\beta)$ *is the number determined by* $\mathbf{A}_i^\alpha \leqq \mathbf{A}_j^\beta$.

6. Example 1

6.1. Exposition of the example

In Section 6 we give an example to demonstrate how the extensive congruences of a pan-similar automaton can be constructed.

Fig. 2 shows the graph of an autonomous automaton **A** (with $|A|=33$ and $|Y|=3$). **A** has two connected components and is pan-similar. The simple homomorphic image of the cycles of **A** can be seen in Fig. 3. For the sake of brevity, we

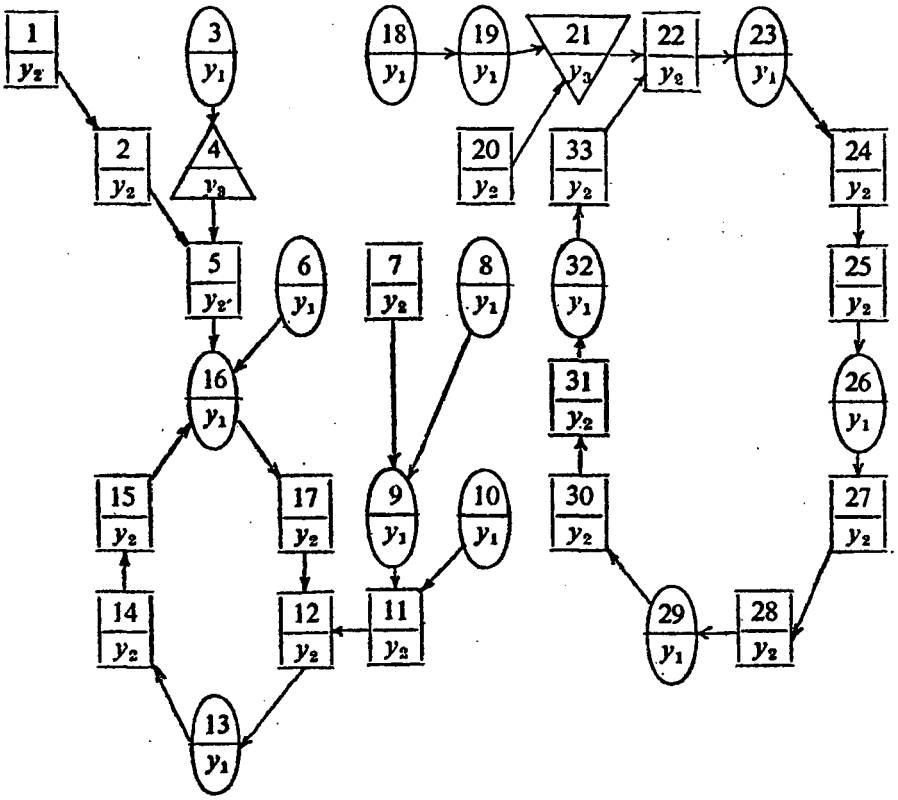


Fig. 2.

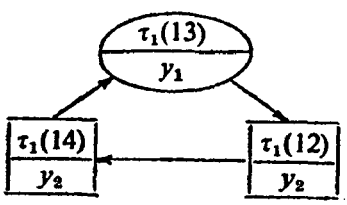


Fig. 3.

Table 1

i	1	2	3	4	5	6	7	8	9	10	11	18	19	20	21
$\sigma(i)$	13	14	13	14	15	15	15	15	16	16	17	31	32	32	33

denote a state simply by i instead of a_i. We make a perspicuous distinction between the output signs y_1, y_2, y_3 so that we draw a circle, a square or a triangle, respectively.

Table 1 shows the values of σ on the acyclic states. D consists of the states 2, 5, 7, 9, 10, 11 and the eighteen cyclic states.

6.2. The realizations of Construction III

The initial step of the construction can be applied in three different ways; we get the sets

$$R_1^{(1)} = \{5\}, \quad R_1^{(2)} = \{11\}, \quad R_1^{(3)} = \{5, 11\}.$$

After an initial step, we have eleven possibilities for applying a general step; the resulting sets are

$$R_2^{(1)} = \{2\}, \quad R_2^{(2)} = \{9\}, \quad R_2^{(3)} = \{10\}, \quad R_2^{(4)} = \{9, 10\}, \quad R_2^{(5)} = \{2, 11\},$$

$$R_2^{(6)} = \{5, 9\}, \quad R_2^{(7)} = \{5, 10\}, \quad R_2^{(8)} = \{5, 9, 10\}, \quad R_2^{(9)} = \{2, 9\},$$

$$R_2^{(10)} = \{2, 10\}, \quad R_2^{(11)} = \{2, 9, 10\}.$$

(If we start with $R_1^{(1)}$, we get $R_2^{(1)}$. The sets $R_2^{(3)}$, $R_2^{(4)}$, $R_2^{(5)}$ are obtained if we start with $R_1^{(2)}$. The remaining seven sets are derived from $R_1^{(3)}$.)

If one of $R_2^{(2)}$, $R_2^{(4)}$, $R_2^{(6)}$, $R_2^{(8)}$, $R_2^{(9)}$, $R_2^{(11)}$ is considered, we can execute a second general step. In this manner we arrive to the following six sets:

$$R_3^{(1)} = \{7\}, \quad R_3^{(2)} = \{7, 10\}, \quad R_3^{(3)} = \{5, 7\},$$

$$R_3^{(4)} = \{5, 7, 10\}, \quad R_3^{(5)} = \{2, 7\}, \quad R_3^{(6)} = \{2, 7, 10\}.$$

We have exhausted all possibilities for performing Construction III. We have got that there are twenty-one choices for the subautomaton occurring in Construction II. (Twenty of these are generated by the constructed sets, respectively; among them, $R_3^{(6)}$ generates the whole sub-automaton **D**. A further subautomaton consists of the cyclic states only.)

6.3. The possibilities for choosing d, z_1, z_2

Now we turn to how Construction II can be performed for the automaton **A**. We have two possibilities for choosing d: either $d=3$ or $d=6$. As we have seen earlier, B can be selected in 21 manners.

If $d=6$, then there are two essentially different[7] possibilities for the choice of the pair $\{z_1, z_2\}$. The first of these is $z_1=12$, $z_2=22$; the other is $z_1=12$, $z_2=25$. If $d=3$, then we have only one possibility (apart from non-essential changes): $z_1=12$, $z_2=22$.

In the previous considerations, we have seen that the number of possibilities for choosing the parameters B, d, z_1, z_2 is 63 $(=21.(2+1))$. In fact, **A** has more than 63 extensive congruences, because Step 4 (III/γ) of Construction II is not strictly determined.

6.4. Some notational conventions

Before dealing with the extensive congruences of **A** in a somewhat (but not fully) detailed manner, it is appropriate to introduce how the partitions of the state set of **A** can be denoted shortly. We agree that, e.g.,

$$\langle 1, 4 \,|\, 2, 3, 11 \,|\, 6, 19 \rangle$$

[7] "Essentially different" is meant in sense of Proposition 2.

denotes the partition in which the three sets $\{1, 4\}$, $\{2, 3, 11\}$, $\{6, 19\}$ are classes and each one of the remaining states forms a one-element class. If it is already known that $H=\{2, 11\}$, then we can write

$$\langle 1, 4 \mid H, 3 \mid 6, 19\rangle$$

instead of the above formula, too.

Let another notation also be introduced in the following way (for sake of conciseness): the formula

$$\langle 1, 8, 11 \mid 3, 9 \parallel (2, 10), (4, 7)\rangle$$

will mean the system consisting of the four partitions

$$\langle 1, 8, 11 \mid 3, 9\rangle,$$
$$\langle 1, 8, 11 \mid 3, 9 \mid 2, 10\rangle,$$
$$\langle 1, 8, 11 \mid 3, 9 \mid 4, 7\rangle,$$
$$\langle 1, 8, 11 \mid 3, 9 \mid 2, 10 \mid 4, 7\rangle.$$

6.5. Study of the extensive congruences obtained through certain subautomata

We have seen in Subsection 6.2 that there are 21 possibilities for choosing $\mathbf{G}_0$. Among these, now we consider the subautomata generated by

$$\emptyset, \quad R_1^{(1)}, \quad R_2^{(3)}, \quad R_2^{(7)}, \quad R_2^{(4)},$$

and we are going to discuss the congruences obtained with these $\mathbf{G}_0$'s. (The discussion of any of the remaining 16 possibilities resembles to one or another of these.)

Introduce the sets (of cyclic states)

$$H_1 = \{12, 15, 22, 25, 28, 31\},$$
$$H_2 = \{13, 16, 23, 26, 29, 32\},$$
$$H_3 = \{14, 17, 24, 27, 30, 33\},$$
$$K_1 = \{12, 22, 28\},$$
$$K_2 = \{13, 23, 29\},$$
$$K_3 = \{14, 24, 30\},$$
$$K_4 = \{15, 25, 31\},$$
$$K_5 = \{16, 26, 32\},$$
$$K_6 = \{17, 27, 33\},$$
$$L_1 = \{12, 25, 31\},$$
$$L_2 = \{13, 26, 32\},$$
$$L_3 = \{14, 27, 33\},$$
$$L_4 = \{15, 22, 28\},$$
$$L_5 = \{16, 23, 29\},$$
$$L_6 = \{17, 24, 30\}.$$

Let us study first the case when $\mathbf{G}_0$ contains the cyclic states only. If $d=3$ (and $z_1=12$, $z_2=22$), then two congruences are obtained with these parameters:

$$\langle H_1|H_2|H_3\|(9, 10)\rangle.$$

Analogously, if $d=6$, $z_1=12$, $z_2=22$, then

$$\langle K_1|K_2|K_3|K_4|K_5|K_6\|(9, 10)\rangle$$

are got; when $d=6$, $z_1=12$, $z_2=25$, then

$$\langle L_1|L_2|L_3|L_4|L_5|L_6\|(9, 10)\rangle$$

are. Altogether, we have obtained six congruences for the smallest possible $\mathbf{G}_0$.

If we start with the subautomaton generated by $R_1^{(1)}$ (as $\mathbf{G}_0$), then we get fourteen congruences

$$\langle H_1, 5|H_2|H_3\|(9, 10)\rangle,$$

$$\langle H_1, 5|H_2|H_3|4, 21\|(3, 19), (9, 10)\rangle,$$

$$\langle K_1|K_2|K_3|K_4, 5|K_5|K_6\|(9, 10)\rangle,$$

$$\langle L_1|L_2|L_3|L_4, 5|L_5|L_6\|(9, 10)\rangle,$$

$$\langle L_1|L_2|L_3|L_4, 5|L_5|L_6|4, 21\|(3, 19), (9, 10)\rangle.$$

With the subautomaton generated by $R_2^{(3)}$, three congruences are obtained:

$$\langle H_1|H_2, 10|H_3, 11\rangle,$$

$$\langle K_1|K_2|K_3|K_4|K_5, 10|K_6, 11\rangle,$$

$$\langle L_1|L_2|L_3|L_4|L_5, 10|L_6, 11\rangle.$$

With the subautomaton generated by $R_2^{(7)}$, we get seven congruences:

$$\langle H_1, 5|H_2, 10|H_3, 11\rangle,$$

$$\langle H_1, 5|H_2, 10|H_3, 11|4, 21\|(3, 19)\rangle,$$

$$\langle K|K_2|K_3|K_4, 5|K_5, 10|K_6, 11\rangle,$$

$$\langle L_1|L_2|L_3|L_4, 5|L_5, 10|L_6, 11\rangle,$$

$$\langle L_1|L_2|L_3|L_4, 5|L_5, 10|L_6, 11|4, 21\|(3, 19)\rangle.$$

Finally, the discussion of the subautomaton generated by $R_2^{(4)}$ leads to twelve congruences:

$$\langle H_1|H_2, 9, 10|H_3, 11|H_4|H_5|H_6\|(5, 7), (6, 8)\rangle,$$

$$\langle K_1|K_2|K_3|K_4|K_5, 9, 10|K_6, 11\|(5, 7), (6, 8)\rangle,$$

$$\langle L_1|L_2|L_3|L_5, 9, 10|L_6, 11\|(5, 7), (6, 8)\rangle.$$

6.6. *Short overview of the extensive congruences of* **A**

Out of the 21 basic sets, five ones were examined in Subsection 6.5. Now we cast a glance to the other 16 ones. The generating sets $R_1^{(2)}$, $R_3^{(1)}$, $R_3^{(2)}$ behave similarly to the smallest $\mathbf{G}_0$ (each of them leads to six congruences). $R_2^{(2)}$ and $R_2^{(10)}$ behave analogously to $R_2^{(4)}$ and $R_2^{(7)}$, respectively. The behaviour of the eleven generating sets not yet mentioned is analogous to $R_1^{(1)}$.

Consequently, the number of extensive congruences of **A** is

$$233 = (4.6 + 2.12 + 2.7 + 12.14 + 1.3).$$

6.7. *Maximal and minimal extensive congruences*

The maximal congruence of **A** is

$$\langle H_1, 5, 7 | H_2, 9, 10 | H_3, 2, 11 | 3, 19 | 4, 21 | 6, 8 \rangle;$$

it can be obtained from $R_3^{(6)}$ and $d=3$.

The question arises whether, for an arbitrary pan-similar automaton, there exists a minimal congruence among the *extensive* ones. The analysis of **A** shows that the answer is negative (in general). Indeed, let the extensive congruences

$$\pi_K = \langle K_1 | K_2 | K_3 | K_4 | K_5 | K_6 \rangle,$$
$$\pi_L = \langle L_1 | L_2 | L_3 | L_4 | L_5 | L_6 \rangle$$

(got with the smallest $\mathbf{G}_0$ and $d=6$) be considered. The system $\{\pi_K, \pi_L\}$ is minimal in the following weak sense: each extensive congruence π satisfies at least one of the relations $\pi_K \leqq \pi$ and $\pi_L \leqq \pi$. None of π_K, π_L is a refinement of the other, their intersection is not extensive.

7. Appendix (Outlook)

7.1. *Theoretical considerations*

Let now $\mathbf{A} = (A, X, Y, \delta, \lambda)$ be an arbitrary (not necessarily autonomous) finite Moore automaton. A partition π of the state set A was called a congruence if $a \equiv b \pmod{\pi}$ implies

$$(\lambda(a) = \lambda(b)) \ \& \ (\delta(a, x) \equiv \delta(b, x) \pmod{\pi}) \tag{7.1}$$

for every choice of $a(\in A), b(\in A)$ and $x(\in X)$ (cf. Section 2). The question to which the present paper is devoted is a particular case of the following general one:

Basic problem. Describe the congruences of an arbitrary automaton **A**.

A satisfactorily explicit solution of this problem is, of course, hopeless in full generality. The importance of the basic problem (in spite of the fact that it seems to be an imaginary question) is that it can be considered as a common source of other problems. More explicitly, it admits several particularizations (into various

directions) so that these particular questions are interesting and their solution lies already (more or less) within the limits of real possibilities. We can pose certain specializations of the basic problem so that one or another of the following constraints is accepted (possibly combined with each other):

(A) A is autonomous, i.e., $|X|=1$.

(B) A is initially connected, i.e., a state $a_0(\in A)$ is distinguished and it is postulated that to each $a(\in A)$ there is an input word p (depending on a) such that $\delta(a_0, p)=a$.

(C) We are not interested in obtaining all congruences of the automata but we want to separate the simple automata from the non-reduced ones. (The results in this direction are considered to be valuable in so far as the method of separation is of constructive character.)

(D) We are not interested in the output function of the automata. (This approach is, strictly spoken, the particular case of the basic problem when we restrict ourselves to the case $|Y|=1$.)

(E) The definition of congruence is strengthened by requiring $\delta(a, x_1)\equiv\delta(b, x_2) \pmod{\pi}$ in the second term of (7.1) ($x_1\in X$, $x_2\in X$). (From a rigidly formal point of view, this is not a particular case of the basic problem. However, this strengthening of the definition implies that the set of congruences of an automaton becomes narrower.)

The specializations (A) and (A) & (E) are the same. If we accept both (D) and (E), we arrive at a purely graph-theoretical problem.

In the paper [4], a (natural and easy) solution of the particular case (A) & (B) & (C) of the basic problem was stated (Section 3) and the constructive aspects of the question were dealt with (Sections 4—5).

In [5], the case (A) & (D) [=(A) & (D) & (E)] was discussed (Chapter II) and these considerations were expanded into an elucidation of the case (D) & (E) for a large class of directed graphs (Chapter III).

In the present paper, a treatment of the case (A) is contained. Thus the theory elaborated now is a common generalization of Section 3 of [4] and Chapter II of [5].

Among the articles whose subject is more or less related to the present paper, let [9], [6], [10] and the most recent publication [7] be mentioned. A number of further references can be found in [10] and [5].

In the author's opinion, the most exciting subproblem of the entire domain of questions is the case (B) & (C). Unfortunately, the topic seems to become terribly more intricate when the autonomousness of the automata is abandoned.

My intention with the papers [2], [3] was that they should be the first steps towards a constructive treatment of the subproblem (B) & (C). As far as it can be predicted, each further step in this direction will require to surmount immense difficulties.

7.2. Examples 2 and 3

Let us finish our paper with two examples which show the difficulties of handling the non-autonomous case.

Statement 1. *Let* $\mathbf{A}=(A, X, Y, \delta, \lambda)$ *be an automaton, consider the* n *autonomous automata* $\mathbf{A}_i=(A, \{x_i\}, Y, \delta_i, \lambda)$ *where* $n=|X|$, x_i *runs through the elements of* X *and* δ_i *is the restriction of* δ *to the case when the second argument is* x_i.

Table 2

a_i	$\delta(a_i, x_1)$	$\delta(a_i, x_2)$	$\lambda(a_i)$
a_1	a_2	a_3	y_1
a_2	a_2	a_4	y_1
a_3	a_5	a_2	y_1
a_4	a_6	a_2	y_1
a_5	a_2	a_2	y_1
a_6	a_2	a_2	y_2

Denote by $\pi_{\max}^{(i)}$ *the maximal congruence of* $\mathbf{A}_i$. *If* $\pi_{\max}^{(1)} \cap \pi_{\max}^{(2)} \cap \ldots \cap \pi_{\max}^{(n)}$ *equals the minimal partition* o *of* A, *then* $\mathbf{A}$ *is simple.*

Statement 1 is almost trivial. It may be asked whether the conversion of (the last sentence of) Statement 1 is valid.

Example 2. Analyze the automaton $\mathbf{A}$ determined by Table 2 (see Fig. 4) (with

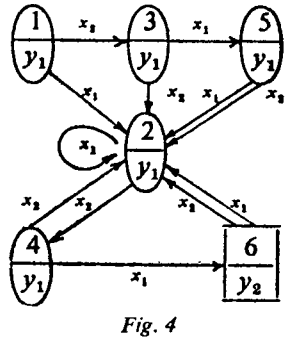


Fig. 4

$n=2$ and $v=|A|=6$). Form the autonomous automata $\mathbf{A}_1$ and $\mathbf{A}_2$. We get that the maximal congruence $\pi_{\max}^{(1)}$ of $\mathbf{A}_1$ is

$$\langle a_1, a_2, a_3, a_5 | a_4 | a_6 \rangle,$$

and the maximal congruence $\pi_{\max}^{(2)}$ of $\mathbf{A}_2$ is

$$\langle a_1, a_2, a_3, a_4, a_5 | a_6 \rangle;$$

hence $\pi_{\max}^{(1)} \cap \pi_{\max}^{(2)} = \pi_{\max}^{(1)}$. On the other hand, the automaton $\mathbf{A}$ itself is reduced.

This means that the condition in Statement 1 is (sufficient but) not necessary for the simplicity. If we take into account the connection between the distinguishability of states and the simplicity[8], then it becomes clear that whenever a pair of different states which are congruent modulo $\pi_{\max}^{(1)} \cap \pi_{\max}^{(2)}$ is considered — e.g., a_1

[8] Cf. [2], Section 5.

and a_2 —, then they are not distinguishable by any word of form x_1^m or x_2^m $(m \geqq 0)$, but there is a "mixed" input word which distinguishes them, for example,

$$\lambda(\delta(a_1, x_2x_1)) = \lambda(a_5) = y_1 \neq y_2 = \lambda(a_6) = \lambda(\delta(a_2, x_2x_1)).$$

Statement 1 has contained a sufficient condition for the *simplicity* of an automaton. The next statement asserts that another condition is sufficient for *non-simplicity*. (We shall see later that also Statement 2 does not allow a conversion.)

Statement 2. *Let* $\mathbf{A}=(A, X, Y, \delta, \lambda)$ *be an automaton, consider two sub-automata* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *of* $\mathbf{A}$.[9] *Suppose that there is an isomorphism* α *of* $\mathbf{A}_1$ *onto* $\mathbf{A}_2$ *such that* α *differs from the identical mapping of the state set of* $\mathbf{A}_1$. *Then* $\mathbf{A}$ *is not reduced.*

Table 3

a_i	$\delta(a_i, x_1)$	$\delta(a_i, x_2)$	$\lambda(a_i)$
a_1	a_2	a_3	y_1
a_2	a_4	a_4	y_2
a_3	a_5	a_5	y_2
a_4	a_6	a_6	y_3
a_5	a_7	a_7	y_3
a_6	a_2	a_1	y_4
a_7	a_3	a_1	y_4

Proof. There is a state a of $\mathbf{A}_1$ such that a and a^α are different. It is easy to see that a and a^α are undistinguishable, hence they are congruent for the maximal congruence of $\mathbf{A}$.

Example 3. Consider the automaton $\mathbf{A}$ determined by Table 3 (see Fig. 5). This automaton has neither a proper sub-automaton nor a non-trivial automorphism.

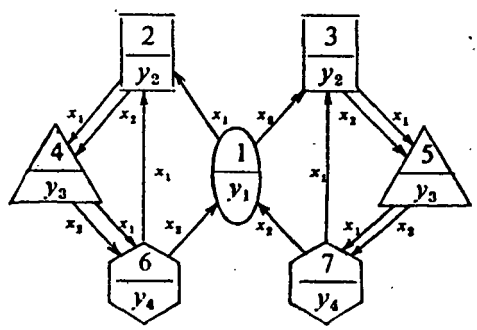


Fig. 5

[9] It is permitted that $\mathbf{A}$, $\mathbf{A}_1$, $\mathbf{A}_2$ be not pairwise different (even all of them can coincide).

Thus the condition of Statement 2 does not apply for **A**. However, **A** is not reduced, its maximal congruence $\pi_{\max}$ is

$$\langle a_1|a_2,\ a_3|a_4,\ a_5|a_6,\ a_7\rangle.$$

Consequently, the (sufficient) condition in Statement 2 is not necessary.

The fact that **A** is not simple but this cannot be shown by use of Statement 2 is in connection with the phenomenon that the *partial* sub-automaton over the state set $\{a_2, a_4, a_6\}$ is isomorphic to the *partial* sub-automaton over $\{a_3, a_5, a_7\}$. It can also be observed that there exists no chain

$$o = \pi_7 \subset \pi_6 \subset \pi_5 \subset \pi_4 = \pi_{\max}$$

in **A** such that π_4, π_5, π_6, π_7 are congruences whose indices (i.e., numbers of classes) are 4, 5, 6, 7, respectively. (Indeed, **A** has no other non-trivial congruence than $\pi_{\max}$.)

MATHEMATICAL INSTITUTE OF THE
HUNGARIAN ACADEMY OF SCIENCES
BUDAPEST 1364, HUNGARY
P.O. BOX 127.

References

[1] ÁDÁM, A., Gráfok és ciklusok (Graphs and cycles), *Mat. Lapok*, v. 22, 1971, pp. 269—282.
[2] ÁDÁM, A., On the question of description of the behaviour of finite automata, *Studia Sci. Math. Hungar.*, v. 13, 1978, pp. 105—124.
[3] ÁDÁM, A., On the complexity of codes and pre-codes assigned to finite Moore automata, *Acta Cybernet.*, v. 5, 1981, pp. 117—133.
[4] ÁDÁM, A., On the simplicity of finite autonomous Moore automata, *Studia Sci. Math. Hungar.*, v. 16, 1981, pp. 427—436.
[5] ÁDÁM, A., On certain partitions of finite directed graphs and of finite automata, *Acta Cybernet.*, v. 6, 1984, pp. 331—346.
[6] BERMAN, J., On the congruence lattices of unary algebras, *Proc. Amer. Math. Soc.*, v. 21, 1972, pp. 34—38.
[7] JAKUBÍKOVÁ-STUDENOVSKÁ, D., On congruence relations of monounary algebras, I, *Czechoslovak Math. J.*, v. 32, 1982, pp. 437—459.
[8] ORE, O., *Theory of graphs*, Amer. Math. Soc., Providence, 1962.
[9] YOELI, M. & GINZBURG, A., On homomorphic images of transition graphs, *J. Franklin Inst.*, v. 278, 1964, pp. 291—296.
[10] (СКОРНЯКОВ, Л. А.) SKORNJAKOV, L. A., Unars, *Universal Algebra, (Colloq. Math. Soc. J. Bolyai, 29, Esztergom, 1977)*, North-Holland, Amsterdam, 1982, pp. 735—743.

(Received July 15, 1984)

On Complexity of Finite Moore Automata

By Masashi Katsura

The concept of complexity of finite Moore automata is introduced by Ádám [1]. In this paper, we obtain relationships among complexity and cardinalities of state set, input set and output set of a Moore automaton.

1.

For a finite set Z, the cardinality of Z is denoted by $|Z|$. Z^* is the free monoid generated by Z. $\mathbf{N}$ is the set of positive integers and $\mathbf{N}^\circ$ is the set of nonnegative integers. For $t, k, \in \mathbf{N}^\circ$, we set $[t: k]=\{i \in \mathbf{N}^\circ | t \leqq i \leqq k\}$.

By a *Moore automaton*, we mean a 5-tuple $\mathbf{A}=(A, X, Y, \delta, \lambda)$, where A, X, Y are finite nonempty sets called a state set, an input set and an output set, respectively. δ is a mapping of $A \times X$ into A called a state transition function (δ is extended as usual to a mapping of $A \times X^*$ into A). λ is a mapping of A *onto* Y called an output function.

Let $\mathbf{A}=(A, X, Y, \delta, \lambda)$ be a Moore automaton. If $\lambda(\delta(a, p)) \neq \lambda(\delta(b, p))$ holds for $a, b \in A$ and $p \in X^*$, then we say that p distinguishes between a and b. $\omega_{\mathbf{A}}(a, b)$ is the minimal length of p which distinguishes between a and b. If there is no word which distinguishes between a and b, then we write $\omega_{\mathbf{A}}(a, b)=\infty$. *The complexity* $\Omega(\mathbf{A})$ of the Moore automaton $\mathbf{A}$ is defined by $\Omega(\mathbf{A})=\max \{\omega_{\mathbf{A}}(a, b) | a, b \in A, a \neq b\}$. If $|A|=1$ then $\Omega(\mathbf{A})=0$.

A Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ is said to be *initially connected* if a distinguished state $a_0 \in A$, called the *initial state* of A, is given and the following condition is satisfied: For any $a \in A$, there exists a $p \in X^*$ such that $\delta(a_0, p)=a$.

Let $v, n \in \mathbf{N}$ and $w \in \mathbf{N}^\circ \cup \{\infty\}$. If there exists an (initially connected) Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ such that $|A|=v$, $|X|=n$ and $\Omega(\mathbf{A})=w$, then the triple (v, n, w) is said to be *realizable* by (initially connected) Moore automata.

We have the following theorem by summarizing the results of Ádám in [2], [3], [4].

Theorem 1. For any $v, n \in \mathbf{N}$ and $w \in \mathbf{N}^\circ \cup \{\infty\}$, the following three statements are equivalent:

(1) (v, n, w) is realizable by Moore automata.

(2) (v, n, w) is realizable by initially connected Moore automata.

(3) (3.1) $w \leqq v-2$, or

(3.2) $v = 1, \quad w = 0,$ or

(3.3) $v \geqq 2, \quad w = \infty.$ □

When we realize a triple (v, n, w) for a small w, a large output set is needed, and vice versa. We wish to take consideration on cardinalities of output sets, too.

Let $v, n, m \in \mathbf{N}$ and $w \in \mathbf{N}^\circ \cup \{\infty\}$. If there exists an (initially connected) Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ such that $|A|=v$, $|X|=n$, $|Y|=m$ and $\Omega(\mathbf{A})=w$, then the 4-tuple (v, n, m, w) is said to be *realizable* by (initially connected) Moore automata.

In this paper, all realizable 4-tuples are completely determined. Section 2 gives a sufficient condition. Section 4 is a preparation to show that the sufficient condition given in Section 2 is necessary. In Section 5, the main result is stated and proved. Section 6 illustrates some examples. In Section 3, we prove a conjecture posed in [3].

2.

Let X and Y be finite nonempty sets and let $t \in \mathbf{N}^\circ$. By $F_t(X, Y)$ we denote the set of all mappings of $\bigcup_{k=0}^{t} X^k$ into Y.

The following lemma is evident.

Lemma 1. $|F_t(X, Y)| = |Y|^{1+|X|+|X|^2+\cdots+|X|^t}$ for any $t \in \mathbf{N}^\circ$. □

Let $\mathbf{A}=(A, X, Y, \delta, \lambda)$ be a Moore automaton. For each $a \in A$, let $\lambda^*(a)$ be a mapping of X^* into Y defined by

$$(\lambda^*(a))(p) = \lambda(\delta(a, p)).$$

For $t \in \mathbf{N}^\circ$, $\lambda^{(t)}(a)$ is an element of $F_t(X, Y)$ which is the restriction of $\lambda^*(a)$ to $\bigcup_{k=0}^{t} X^k$. Hence $\lambda^{(0)}=\lambda$ if we identify $F_0(X, Y)$ with Y.

For each $t \in \mathbf{N}^\circ$, let $\eta_t(\mathbf{A})$ be a partition of A defined as follows: a and b are congruent modulo $\eta_t(\mathbf{A})$ iff $\omega_{\mathbf{A}}(a, b) \geqq t$. $\eta_t(\mathbf{A})$ is introduced and investigated in [2], [4]. The number of $\eta_t(\mathbf{A})$-classes is denoted by $|\eta_t(\mathbf{A})|$. The following three lemmas indicate fundamental properties of the partition $\eta_t(\mathbf{A})$.

Lemma 2 [4]. $\eta_0(\mathbf{A}) \supseteq \eta_1(\mathbf{A}) \supseteq \eta_2(\mathbf{A}) \supseteq \cdots$. □

Lemma 3 [4]. If $\eta_{t-1}(\mathbf{A})=\eta_t(\mathbf{A})$ then $\eta_t(\mathbf{A})=\eta_{t+1}(\mathbf{A})$. □

Lemma 4. Let $w \in \mathbf{N}^\circ$. Then $\Omega(\mathbf{A})=w$ iff $\eta_w(\mathbf{A}) \supsetneqq \eta_{w+1}(\mathbf{A})$ and $|\eta_{w+1}(\mathbf{A})| = |A|$. □

By using the mapping $\lambda^{(t)}(a)$, the partition $\eta_t(A)$ is characterized as follows:

Lemma 5. a and b are congruent modulo $\eta_{t+1}(\mathbf{A})$ iff $\lambda^{(t)}(a)=\lambda^{(t)}(b)$. □

Hence we have:

Lemma 6. $|\eta_{t+1}(\mathbf{A})| = |\{\lambda^{(t)}(a) \in F_t(X, Y) | a \in A\}|$ for any $t \in \mathbf{N}^\circ$. □

Expecially, we have:

Lemma 7. $|\eta_{t+1}(\mathbf{A})| \leqq |Y|^{1+|X|+|X|^2+\cdots+|X|^t}$ for any $t \in \mathbf{N}^\circ$. □

On the other hand, a lower bound of the number of $\eta_t(\mathbf{A})$-classes is given as follows:

Lemma 8. Let $t \in \mathbf{N}^\circ$. If $t \leqq \Omega(\mathbf{A})$ then $|\eta_{t+1}(\mathbf{A})| \geqq |Y|+t$.

Proof. By Lemmas 2, 3 and 4, we have $\eta_1(\mathbf{A}) \supsetneqq \eta_2(\mathbf{A}) \supsetneqq \ldots \supsetneqq \eta_t(\mathbf{A}) \supsetneqq \eta_{t+1}(\mathbf{A})$, i.e., $|\eta_1(\mathbf{A})| < |\eta_2(\mathbf{A})| < \ldots < |\eta_t(\mathbf{A})| < |\eta_{t+1}(\mathbf{A})|$. Hence $|\eta_{t+1}(\mathbf{A})| \geqq |\eta_1(\mathbf{A})|+t$. By Lemma 6, we have $|\eta_1(\mathbf{A})| = |Y|$. □

Now, we have the following desired result.

Proposition 1. Let $v, n, m \in \mathbf{N}$ and $w \in \mathbf{N}^\circ$. If the 4-tuple (v, n, m, w) is realizable by Moore automata then $m+w \leqq v \leqq m^{1+n+n^2+\cdots+n^w}$.

Proof. Let $\mathbf{A}=(A, X, Y, \delta, \lambda)$ be a Moore automaton such that $|A|=v$, $|X|=n$, $|Y|=m$ and $\Omega(\mathbf{A})=w$. By Lemma 4, $|\eta_{w+1}(\mathbf{A})|=v$. By Lemmas 7 and 8, we have $m+w \leqq |\eta_{w+1}(\mathbf{A})| \leqq m^{1+n+n^2+\cdots+n^w}$. □

3.

Ádám posed three conjectures in [3]. Conjectures 1 and 2 are solved in Theorem 1. However, Conjecture 3 is not yet solved. In this section, we settle this conjecture. (This result is not used in what follows).

Let $\mathbf{A}=(A, X, Y, \delta, \lambda)$ be a Moore automaton such that $1 \leqq \Omega(\mathbf{A}) < \infty$. Put $\Omega(\mathbf{A})=w$. Take $a, b \in A$ such that $\omega_{\mathbf{A}}(a, b)=w$. Then there exists a $q \in X^w$ such that $\lambda(\delta(a, q)) \neq \lambda(\delta(b, q))$. Let $q=q'x$ with $q' \in X^{w-1}$ and $x \in X$. Let B be the $\eta_2(\mathbf{A})$-class containing $\delta(a, q')$, i.e., $B=\{c \in A \mid \lambda(\delta(c, p))=\lambda(\delta(a, q'p))$ for any $p \in X \cup \{e\}\}$, where e is the identity of X^*.

Define $\mathbf{A}'=(A, X, Y', \delta, \lambda')$ as follows:

(i) $Y'=Y \cup \{y\}$ where y is not in Y.
(ii) $\lambda'(c)=y$ for any $c \in B$.
(iii) $\lambda'(c)=\lambda(c)$ for any $c \in A-B$.

Since $\lambda(\delta(a, q'x)) \neq \lambda(\delta(b, q'x))$, we have $\delta(b, q') \notin B$. Hence $\lambda'(\delta(b, q')) = \lambda(\delta(b, q')) = \lambda(\delta(a, q'))$. Consequently, λ' is surjective. Moreover, we have:

Lemma 9. (1) $\eta_t(\mathbf{A}) \supseteq \eta_t(\mathbf{A}')$ for any $t \in \mathbf{N}^\circ$.
(2) $\eta_w(\mathbf{A}) \supsetneqq \eta_w(\mathbf{A}')$.
(3) $\eta_{w-1}(\mathbf{A}')$ is not the identity partition.

Proof. (1) It is obvious that for any $c, d \in A$, if $\lambda(c) \neq \lambda(d)$, then $\lambda'(c) \neq \lambda'(d)$. It follows from this fact that $\omega_{\mathbf{A}'}(c, d) \leqq \omega_{\mathbf{A}}(c, d)$ for any $c, d \in A$.

(2) Since $\lambda'(\delta(a, q')) \neq \lambda(\delta(a, q')) = \lambda(\delta(b, q')) = \lambda'(\delta(b, q'))$, we have $\omega_{\mathbf{A}'}(a, b) \leqq w-1$. Hence a and b are congruent modulo $\eta_w(\mathbf{A})$, but not congruent modulo $\eta_w(\mathbf{A}')$.

(3) It suffices to show that $\omega_{\mathbf{A}'}(a, b)=w-1$. When $w=1$, the conclusion is obvious. Assume $w \geqq 2$ and $\omega_{\mathbf{A}'}(a, b) \leqq w-2$. Then there exists a $p \in \bigcup_{k=0}^{w-2} X^k$ such that $\lambda'(\delta(a, p)) \neq \lambda'(\delta(b, p))$. Since $\lambda(\delta(a, p)) = \lambda(\delta(b, p))$, we have $\delta(a, p) \in B$

and $\delta(b,p)\notin B$, or vice versa. In other words, $\delta(a,p)$ and $\delta(b,p)$ are not congruent modulo $\eta_2(A)$. For any $p'\in X\cup\{e\}$, we have $pp'\in\bigcup_{k=0}^{w-1}X^k$. Hence $\lambda(\delta(a,pp'))=\lambda(\delta(b,pp'))$ for any $p'\in X\cup\{e\}$. This means that $\delta(a,p)$ and $\delta(b,p)$ are congruent modulo $\eta_2(\mathbf{A})$. This is a contradiction. Hence we have $\omega_{\mathbf{A}'}(a,b)=w-1$. □

Proposition 2 ([3] Conjecture 3). Let $\mathbf{A}=(A,X,Y,\delta,\lambda)$ be a Moore automaton such that $1\leqq\Omega(\mathbf{A})<\infty$. Then there exists a Moore automaton $\mathbf{A}'=(A,X,Y',\delta,\lambda')$ such that $|Y'|=|Y|+1$ and $\Omega(\mathbf{A})-1\leqq\Omega(\mathbf{A}')\leqq\Omega(\mathbf{A})$.

Proof. Let $\mathbf{A}'$ be the Moore automaton constructed as above. Lemma 9 (1) implies that $\Omega(\mathbf{A}')\leqq\Omega(\mathbf{A})$. Lemma 9 (3) means that $\Omega(\mathbf{A}')\geqq\Omega(\mathbf{A})-1$. □

As pointed out in [3], we get an automaton of complexity 0 by at most $|A|-|Y|$ times application of Proposition 2. Hence we have another proof of the left hand side inequality of Proposition 1.

4.

In this section, we prepare for showing the converse of Proposition 1. Throughout this section, we assume that $X=\{x_1,\ldots,x_n\}$, $Y=\{y_1,\ldots,y_m\}$ and $m\geqq 2$. $F_t(X,Y)$ is simply denoted by F_t. F_{-1} means the singleton set consisting of the empty mapping, i.e., the mapping whose definition domain is the empty set.

Let $t\in\mathbf{N}^\circ$ and $f\in F_t$. We define $f_l, f_{r,1},\ldots,f_{r,n}\in F_{t-1}$ as follows:

$$f_l \text{ is the restriction of } f \text{ to } \bigcup_{k=0}^{t-1}X^k,$$

$$f_{r,j}(p)=f(x_jp) \text{ for any } p\in\bigcup_{k=0}^{t-1}X^k \quad (j\in[1:n]).$$

Hence for $f\in F_0$, f_l and $f_{r,j}$ are the empty mappings. f_l is said to be the *left factor* of f, and $f_{r,j}$ is its *j-th right factor*.

The following lemma can be shown by a straightforward verification.

Lemma 10. Let $t\in\mathbf{N}^\circ$ and $g\in F_{t-1}$. Then $|\{f\in F_t|f_l=g\}|=|\{f\in F_t|f_{r,j}=g\}|=|F_t|/|F_{t-1}|=m^{n^t}$. □

Let $\mathbf{A}=(A,X,Y,\delta,\lambda)$ be a Moore automaton. Consider the mapping $\lambda^{(t)}$ of A into F_t. The assumption that λ is surjective is equivalent to:

(i) For any $j\in[1:n]$, there exists an $a\in A$ such that $(\lambda^{(t)}(a))(e)=y_j$, where e is the identity of X^*.

If $\delta(a,x_j)=b$, then $(\lambda^{(t)}(a))_{r,j}=(\lambda^{(t)}(b))_l$. Hence we have:

(ii) For any $a\in A$ and $j\in[1:n]$, there exists a $b\in A$ such that $(\lambda^{(t)}(a))_{r,j}=(\lambda^{(t)}(b))_l$.

$\Omega(\mathbf{A})=t$ is equivalent to:

(iii) $\lambda^{(t)}$ is injective, and $(\lambda^{(t)}(a))_l=(\lambda^{(t)}(b))_l$ for some $a,b\in A$ with $a\neq b$.

Conversely, assume that a mapping τ of A into F_t which satisfies (i) and (ii) is given. Define a Moore automaton $\mathbf{A}_\tau=(A,X,Y,\delta,\lambda)$ as follows:

(iv) $\lambda(a)=(\tau(a))(e)$ for any $a\in A$.

(v) Let $a\in A$ and $j\in[1:n]$. By (ii), there exists a $b\in A$ such that $\tau(a)_{r,j}=\tau(b)_l$. Set $\delta(a, x_j)=b$.

Then it can easily be seen that $\lambda^{(t)}(a)=\tau(a)$ holds for any $a\in A$. $\mathbf{A}_\tau$ is not unique in general. The collection of all $\mathbf{A}_\tau$ coincides with all Moore automata $\mathbf{A}=(A, X, Y, \delta, \lambda)$ which satisfy $\lambda^{(t)}=\tau$. The partitions $\eta_0(\mathbf{A}_\tau), \eta_1(\mathbf{A}_\tau), \ldots, \eta_{t+1}(\mathbf{A}_\tau)$ are independent of the choice of b in (v), i.e., they depend only on the mapping τ.

To show the converse of Proposition 1, it suffices to give a mapping τ of A into F_t which satisfies (i), (ii) and (iii) for each finite set A with $m+t\leqq|A|\leqq m^{1+n+n^2+\cdots+n^t}$. However, we wish to prove the converse of Proposition 1 in case of initially connected Moore automata. Related problem is:

Let τ be a mapping satisfying (i) and (ii) (and (iii)). What conditions are required so that we can make $\mathbf{A}_\tau$ to be initially connected? What is a method to construct an initially connected $\mathbf{A}_\tau$, when it exists?

In general, this problem seems to be difficult. In what follows, we construct a special type of mapping τ, and construct a special type of initially connected Moore automaton $\mathbf{A}_\tau$.

Let $t\in\mathbf{N}^\circ$, $s\in\mathbf{N}$ and let π be an injection of $[1:s]$ into F_t. If the following four conditions are satisfied then π is called an *op-mapping* of *degree* (t, s) (with respect to X and Y).

(a) For any $g\in F_{t-1}$, there exists an $i\in[1:s]$ such that $\pi(i)_l=g$.

(b) For any $j\in[1:m]$, there exists an $i\in[1:s]$ such that $(\pi(i))(e)=y_j$.

(c) $\pi(i)_{r,1}=\pi(i+1)_l$ for any $i\in[1:s-1]$.

(d) There exists an $i_\pi\in[1:s-1]$ such that $(\pi(i_\pi))(p)=y_1$ for any $p\in\bigcup_{k=0}^{t} X^k$.

(Since π is injective, i_π is uniquely determined).

When $t\geqq 1$, the assertion (b) is implied by (a). When $t=0$, the assertions (a) and (c) are always satisfied, and the assertion (b) means that π is surjective. Hence an op-mapping π of degree $(0, s)$ is considered as a bijection of $[1:s]$ onto Y such that $\pi(i)=y_1$ for some $i\in[1:s-1]$. Thus we have:

Lemma 11. There exists an op-mapping of degree $(0, s)$ iff $s=m$. □

Lemma 12. Let $t, s\in\mathbf{N}$. If there exists an op-mapping π of degree (t, s) then $m^{1+n+n^2+\cdots+n^{t-1}}+1\leqq s\leqq m^{1+n+n^2+\cdots+n^t}$.

Proof. Since π is injective, we have $s\leqq|F_t|$. We have $\pi(i_\pi)_l=\pi(i_\pi)_{r,1}$. Since $i_\pi\in[1:s-1]$, we have $i_\pi+1\in[1:s]$ and $\pi(i_\pi+1)_l=\pi(i_\pi)_{r,1}=\pi(i_\pi)_l$. From this fact and by the assertion (a), it follows that $s\geqq|F_{t-1}|+1$. □

Now we shall construct an op-mapping of degree (t, s) for any $t, s\in\mathbf{N}$ with $m^{1+n+n^2+\cdots+n^{t-1}}+1\leqq s\leqq m^{1+n+n^2+\cdots+n^t}$. To this end, we provide the following two lemmas.

Lemma 13. Let π be an op-mapping of degree (t, s). Then the following statements are equivalent:

(1) There exists an op-mapping π' of degree $(t, s+1)$ which is an extension of π.

(2) There exists an $f\in F_t-\{\pi(i)\mid i\in[1:s]\}$ such that $f_l=\pi(s)_{r,1}$.

Proof. (1)⇒(2). By the assertion (c), we have $\pi'(s+1)_l=\pi'(s)_{r,1}=\pi(s)_{r,1}$. Since π' is injective, $\pi'(s+1)\in F_t-\{\pi(i)\,|\,i\in[1:s]\}$.
(2)⇒(1). Let $\pi'(s+1)=f$ and $\pi'(i)=\pi(i)$ for any $i\in[1:s]$. Then π' is an op-mapping of degree $(t, s+1)$. □

Lemma 14. Let π be an op-mapping of degree (t, s). Assume that there exists no op-mapping π' of degree $(t, s+1)$ which is an extension of π. Then $\pi(s)_{r,1}=\pi(1)_l$.

Proof. If $t=0$ then $\pi(s)_{r,1}$ and $\pi(1)_l$ are the empty mappings. Hence the conclusion holds obviously. Assume that $t\in\mathbf{N}$. Let

$$I=\{i\in[1:s]\,|\,\pi(i)_l=\pi(s)_{r,1}\} \quad \text{and}$$

$$J=\{j\in[1:s]\,|\,\pi(j)_{r,1}=\pi(s)_{r,1}\}.$$

Suppose that $|I|<m^{n^t}$. Then, by Lemma 10, there exists an $f\in F_t-\{\pi(i)|i\in[1:s]\}$ such that $f_l=\pi(s)_{r,1}$. It contradicts the assumption by Lemma 13. Hence we have $|I|=m^{n^t}$. By Lemma 10, we have $|J|\leqq|I|$. By the assertion (c), we have:

If $i\in I-\{1\}$, then $i-1\in J$.

If $j\in J-\{s\}$, then $j+1\in I$.

Hence $|I-\{1\}|=|J-\{s\}|$. Since $s\in J$, we have

$$|I|\geqq|J|=|J-\{s\}|+1=|I-\{1\}|+1.$$

Thus, we have $1\in I$, i.e., $\pi(1)_l=\pi(s)_{r,1}$. □

There exists an op-mapping of degree $(0, m)$ (Lemma 11). Hence to construct an op-mapping of degree (t, s) for each $t,s\in\mathbf{N}$ with $m^{1+n+n^2+\cdots+n^{t-1}}+1\leqq s\leqq m^{1+n+n^2+\cdots+n^t}$, it suffices to give construction methods for the following two cases:

(I) Let $t\in\mathbf{N}$ and $s=m^{1+n+n^2+\cdots+n^{t-1}}+1$. Assume that an op-mapping π of degree $(t-1, s-1)$ is given. Construct an op-mapping π' of degree (t, s).

(II) Let $t, s\in\mathbf{N}$ and $m^{1+n+n^2+\cdots+n^{t-1}}+2\leqq s\leqq m^{1+n+n^2+\cdots+n^t}$. Assume that an op-mapping π of degree $(t, s-1)$ is given. Construct an op-mapping π' of degree (t, s).

Case (II) is divided into the following two subcases:

(II.1) There exists an $f\in F_t-\{\pi(i)\,|\,i\in[1:s-1]\}$ such that $f_l=\pi(s-1)_{r,1}$.
(II.2) There exists no $f\in F_t-\{\pi(i)\,|\,i\in[1:s-1]\}$ such that $f_l=\pi(s-1)_{r,1}$.

Construction (I). (i) Define a mapping σ of $[2:s]$ into F_{t-1} as follows:

$$\sigma(2)=\pi(i_\pi),\ \sigma(3)=\pi(i_\pi+1),\ \ldots,\ \sigma(s-i_\pi+1)=\pi(s-1),$$

$$\sigma(s-i_\pi+2)=\pi(1),\ \sigma(s-i_\pi+3)=\pi(2),\ \ldots,\ \sigma(s)=\pi(i_\pi-1)$$

where i_π is determined in the assertion (d). (By Lemmas 11 and 12, π satisfies the assumption of Lemma 14. Hence $\pi(s-1)_{r,1}=\pi(1)_l$. Thus, we have $\sigma(i)_{r,1}=\sigma(i+1)_l$ for any $i\in[2:s-1]$.)
(ii) Define a mapping π' of $[1:s]$ into F_t as follows:

$$(\pi'(1))(p)=y_1 \quad \text{for any } p\in\bigcup_{k=0}^{t}X^k.$$

Let $i\in[2: s-1]$. Take an $f\in F_t$ such that $f_l=\sigma(i)$ and $f_{r,1}=\sigma(i+1)$. (The existence of such an f follows from $\sigma(i)_{r,1}=\sigma(i+1)_l$). Set $\pi'(i)=f$.

Take an $f\in F_t$ such that $f_l=\sigma(s)$. (The existence of such an f is evident.) Set $\pi'(s)=f$.

Then it is not difficult to verify that π' is an op-mapping of degree (t, s).

Construction (II.1). Take an $f\in F_t-\{\pi(i)\mid i\in[1: s-1]\}$ such that $f_l=\pi(s-1)_{r,1}$. Set $\pi'(s)=f$ and $\pi'(i)=\pi(i)$ for any $i\in[1: s-1]$. Then $g\pi'$ is an op-mappin of degree (t, s).

Construction (II.2). (i) Take an $f\in F_t-\{\pi(i)\mid i\in[1: s-1]\}$.
(ii) Take an $i_0\in[1: s-1]$ such that $f_l=\pi(i_0)_l$. (The existence of such an i_0 follows from the assertion (a). If $i_0=1$, then $f_l=\pi(1)_l=\pi(s-1)_{r,1}$ which contradicts the assumption. Hence we have $i_0\in[2: s-1]$.)
(iii) Define a mapping π' of $[1: s]$ into F_t by

$$\pi'(1)=\pi(i_0),\ \pi'(2)=\pi(i_0+1),\ \ldots,\ \pi'(s-i_0)=\pi(s-1),$$

$$\pi'(s-i_0+1)=\pi(1),\ \ldots,\ \pi'(s-1)=\pi(i_0-1)\quad\text{and}\quad\pi'(s)=f.$$

By Lemmas 13 and 14, we have $\pi'(s-i_0)_{r,1}=\pi(s-1)_{r,1}=\pi(1)_l=\pi'(s-i_0+1)_l$. By the assertion (c), we have $\pi'(s-1)_{r,1}=\pi(i_0-1)_{r,1}=\pi(i_0)_l=f_l=\pi'(s)_l$. It can easily be seen that π' satisfies the other assertions for an op-mapping of degree (t, s).

We have shown the following.

Proposition 3. Let $t, s\in\mathbf{N}$. Then there exists an op-mapping of degree (t, s) iff $m^{1+n+n^2+\cdots+n^{t-1}}+1\leqq s\leqq m^{1+n+n^2+\cdots+n^t}$. □

Remark. Ito and Duske [5] shows the following:

Let Y be a finite nonempty set and let $t\in\mathbf{N}$. Then there exists a $p\in Y^*$ whose length is $|Y|^t+t-1$, and which contains every element of Y^t as a subword (such a word p is called a *merged word* of Y^t).

With a little change of the proof, we have Proposition 3 in case $|X|=1$. Our above constructions are done along the line of Ito and Duske. □

Let π be an op-mapping of degree (t, s) and let $r\in\mathbf{N}^\circ$. Define an automaton $\mathbf{A}(\pi, r)=(A, X, Y, \delta, \lambda)$ as follows:

(e) $A=\{a_1, \ldots, a_s, b_1, \ldots, b_r\}$. Put $b_0=a_{i_\pi}$ and $b_{r+1}=a_{i_\pi+1}$.
(f) $\lambda(a_i)=(\pi(i))(e)$ for any $i\in[1: s]$.
(g) $\lambda(b_i)=y_1$ for any $i\in[1: r]$.
(h) $\delta(a_i, x_1)=a_{i+1}$ for any $i\in[1: s-1]-\{i_\pi\}$.
(i) $\delta(b_i, x_j)=b_{i+1}$ for any $i\in[0: r]$ and $j\in[1: n]$.
(j) Let $(i, j)\in([1: s]-\{i_\pi\})\times[2: n]\cup\{(s, 1)\}$. By the assertion (a), there exists a $k\in[1: s]$ such that $\pi(i)_{r,j}=\pi(k)_l$. Set $\delta(a_i, x_j)=a_k$.

$\mathbf{A}(\pi, r)$ is not unique in general. (If we take the least k in (j), then $\mathbf{A}(\pi, r)$ is uniquely determined.)

It follows from the assertions (b) and (f) that λ is surjective. For any $c\in A$, there exists a $u\in\mathbf{N}^\circ$ such that $\delta(a_1, x_1^u)=c$. Hence $\mathbf{A}(\pi, r)$ is an initially connected Moore automaton with initial state a_1.

Lemma 15. $\lambda^{(t)}(a_i)=\pi(i)$ for any $i\in[1:s]$, and $(\lambda^{(t)}(b_i))(p)=y_1$ for any $i\in[1:r]$ and $p\in\bigcup_{k=0}^{t} X^k$.

Proof. For each: $u\in[0:t]$, we consider the following two conditions:

$(\mathscr{C}_u)$ $(\lambda^{(t)}(a_i))(p)=(\pi(i))(p)$ for any $i\in[1:s]-\{i_\pi\}$ and $p\in\bigcup_{k=0}^{u} X^k$.

$(\mathscr{D}_u)$ $(\lambda^{(t)}(b_i))(p)=y_1$ for any $i\in[0:r]$ and $p\in\bigcup_{k=0}^{u} X^k$.

$(\mathscr{C}_0)$ and $(\mathscr{D}_0)$ follow directly from (f) and (g). Let $u\in[1:t]$ and assume that $(\mathscr{C}_{u-1})$, $(\mathscr{D}_{u-1})$ hold. Let $p\in X^u$. Then $p=x_jq$ for some $j\in[1:n]$ and $q\in X^{u-1}$. Let $i\in[1:s]-\{i_\pi\}$ and $\delta(a_i, x_j)=a_k$. Then $\pi(k)_l=\pi(i)_{r,j}$ by (h), (c) and (j). We have

$$(\lambda^{(t)}(a_i))(p)=\lambda(\delta(a_i, p))=\lambda(\delta(a_k, q))=(\lambda^{(t)}(a_k))(q)=(\pi(k)_l)(q)=(\pi(i)_{r,j})(q)=$$
$$=(\pi(i))(x_jq)=(\pi(i))(p).$$

Hence we have $(\mathscr{C}_u)$. Let $i\in[0:r]$. Then $\lambda(\delta(b_i, p))=\lambda(\delta(b_{i+1}, q))=$ $=(\lambda^{(t)}(b_{i+1}))(q)=y_1$. Hence we have $(\mathscr{D}_u)$. Consequently, we have $(\mathscr{C}_t)$ and $(\mathscr{D}_t)$ by induction. □

Lemma 16. Let π be an op-mapping of degree (t, s) and let $r\in\mathbf{N}^\circ$. For a Moore automaton $\mathbf{A}(\pi, r)=(A, X, Y, \delta, \lambda)$, we have:

$$|\eta_0(\mathbf{A}(\pi, r))| = 1,$$
$$|\eta_1(\mathbf{A}(\pi, r))| = m,$$
$$|\eta_2(\mathbf{A}(\pi, r))| = m^{1+n},$$
$$\cdots$$
$$|\eta_t(\mathbf{A}(\pi, r))| = m^{1+n+n^2+\ldots+n^{t-1}},$$
$$|\eta_{t+1}(\mathbf{A}(\pi, r))| = s = |A|-r,$$
$$|\eta_{t+2}(\mathbf{A}(\pi, r))| = |A|-(r-1),$$
$$\cdots$$
$$|\eta_{t+r}(\mathbf{A}(\pi,r))| = |A|-1,$$
$$|\eta_{t+r+1}(\mathbf{A}(\pi,r))| = |A|.$$

Proof. $|\eta_0(\mathbf{A}(\pi, r))|=1$ is evident. Let $u\in[1:t]$. By the assertion (a) and by Lemma 15, for any $g\in F_{u-1}$, there exists an $i\in[1:s]$ such that $\lambda^{(u-1)}(a_i)=g$. Hence by Lemmas 6 and 1, we have

$$|\eta_u(\mathbf{A}(\pi, r))| = |F_{u-1}| = m^{1+n+n^2+\ldots+n^{u-1}}.$$

Since π is injective, it follows from Lemmas 15 and 5 that any two elements of $\{a_1, \ldots, a_s\}$ are not congruent modulo $\eta_{t+1}(\mathbf{A}(\pi, r))$. Moreover by Lemmas 15 and 5, any two elements of $\{b_0, b_1, \ldots, b_r\}$ are congruent modulo $\eta_{t+1}(\mathbf{A}(\pi, r))$. Thus we have $|\eta_{t+1}(\mathbf{A}(\pi, r))|=s$.

Next let $u \in [2: r+1]$. By the assertions (c) and (d), we have $(\pi(i_\pi+1))(p)=y_1$ for any $p \in \bigcup_{k=0}^{t-1} X^k$. Since an op-mapping is injective, we have $\pi(i_\pi+1) \neq \pi(i_\pi)$. Hence $(\pi(i_\pi+1))(q) \neq y_1$ for some $q \in X^t$. Notice that $b_{r+1}=a_{i_\pi+1}$. By the first part of Lemma 15, we have $\lambda(\delta(b_{r+1}, p))=(\lambda^{(t)}(b_{r+1}))(p)=\pi(i_\pi+1)(p)=y_1$ for any $p \in \bigcup_{k=0}^{t-1} X^k$, and $\lambda(\delta(b_{r+1}, q))=(\lambda^{(t)}(b_{r+1}))(q)=\pi(i_\pi+1)(q) \neq y_1$ for some $q \in X^t$. Hence for any $i \in [0: r+1]$, $\lambda(\delta(b_i, p'))=y_1$ for any $p' \in \bigcup_{k=0}^{t+r-i} X^k$ and $\lambda(\delta(b_i, q')) \neq y_1$ for some $q' \in X^{t+r+1-i}$. It follows easily from this fact that $\{b_0, \ldots, b_{r+1-u}\}$ is an $\eta_{t+u}(\mathbf{A}(\pi, r))$-class, and any other element of A is congruent only to itself. Hence we have $|\eta_{t+u}(\mathbf{A}(\pi, r))|=|A|-(r+1-u)$. □

Proposition 4. Let π be an op-mapping of degree (t, s) and let $r \in \mathbf{N}^\circ$. Then $\mathbf{A}(\pi, r)$ is an initially connected Moore automaton with $\Omega(\mathbf{A}(\pi, r))=t+r$.

Proof. By Lemmas 16 and 4. □

Remark. By Lemmas 7, 8 and 16 we have the following: For every $i \in \mathbf{N}^\circ$, the number of $\eta_i(\mathbf{A}(\pi, r))$-classes takes the maximal value among all Moore automata $\mathbf{A}=(A, X, Y, \delta', \lambda')$ with $\Omega(\mathbf{A})=r+t$. □

5.

Now we can determine all realizable 4-tuples.

Theorem 2. Let $v, n, m \in \mathbf{N}$ and $w \in \mathbf{N}^\circ \cup \{\infty\}$. The following three assertions are equivalent:

(1) (v, n, m, w) is realizable by Moore automata.

(2) (v, n, m, w) is realizable by initially connected Moore automata.

(3) (3.1) $m+w \leqq v \leqq m^{1+n+n^2+\ldots+n^w}$, or

(3.2) $w=\infty$, $m \leqq v-1$.

Proof. (2)⇒(1). Obvious.

(1)⇒(3). If $w<\infty$, then we have (3.1) by Proposition 1. If $|A|=|Y|$ in a Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$, then it is evident that $\Omega(\mathbf{A})=0$. Hence we have (3.2).

(3.1)⇒(2). If $m=1$ then (3.1) implies that $v=1$ and $w=0$. For any $n \in \mathbf{N}$, there actually exists a Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ such that $|A|=|Y|=1$ and $|X|=n$. Obviously, $\mathbf{A}$ is initially connected and $\Omega(\mathbf{A})=0$.

Next assume that $m \geqq 2$. Put $\alpha(-1)=m-1$ and $\alpha(k)=m^{1+n+n^2+\cdots+n^k}-k$ for any $k \in \mathbf{N}^\circ$. Our assumption is

(i) $m \leqq v-w \leqq \alpha(w)$.

Since $m=\alpha(0)<\alpha(1)<\alpha(2)<\ldots$, there exists a unique $t \in \mathbf{N}^\circ$ such that

(ii) $\alpha(t-1)+1 \leqq v-w \leqq \alpha(t)$.

Let $X=\{x_1, \ldots, x_n\}$ and $Y=\{y_1, \ldots, y_m\}$. If $t=0$ then (ii) means that $v-w=m$. Hence $(t, v-w+t)=(0, m)$. By Lemma 11, there exists an op-mapping π of degree $(t, v-w+t)$ with respect to X and Y. If $t\geqq 1$ then (ii) means that

$$m^{1+n+n^2+\ldots+n^{t-1}}+2 \leqq v-w+t \leqq m^{1+n+n^2+\ldots+n^t}.$$

Hence by Proposition 3, there exists an op-mapping π of degree $(t, v-w+t)$ with respect to X and Y. By (i) and (ii), it can easily be seen that $t\leqq w$. Consider an initially connected Moore automaton $\mathbf{A}(\pi, w-t)=(A, X, Y, \delta, \lambda)$. We have $|A|=(v-w+t)+(w-t)=v$, $|X|=n$, $|Y|=m$ and, by Proposition 4, $\Omega(\mathbf{A}(\pi, w-t))=t+(w-t)=w$.

(3.2)$\Rightarrow$(2). Define a Moore automaton $\mathbf{A}=(A, X, Y, \delta, \lambda)$ as follows:

(i) $A=\{a_1, \ldots, a_v\}$, $X=\{x_1, \ldots, x_n\}$ and $Y=\{y_1, \ldots, y_m\}$.

(ii) $\lambda(a_i)=y_i (i\in[1: m-1])$ and $\lambda(a_i)=y_m (i\in[m: v])$.

(iii) $\delta(a_i, x_j)=a_{i+1} (i\in[1: v-1])$ and $\delta(a_v, x_j)=a_v$ for any $j\in[1: n]$.

Then it can easily be seen that $\mathbf{A}$ is initially connected and $\omega_{\mathbf{A}}(a_{v-1}, a_v)=\infty$. Hence $\Omega(\mathbf{A})=\infty$, and thus we have (2). □

6.

Let $X=\{x_1, x_2\}$ and $Y=\{1, 2\}$. (Instead of $Y=\{y_1, y_2\}$, we use $Y=\{1, 2\}$ for simplicity). We shall construct op-mappings according to the Constructions (I) and (II) in Section 4. An op-mapping of degree (t, s) is denoted by $\pi_{t,s}$.

For $t=0$, $\pi_{0,2}$ is uniquely determined by $(\pi_{0,2}(1))(e)=1$ and $(\pi_{0,2}(2))(e)=2$. For $t=1$, $\pi_{1,s}$ exist for $2+1\leqq s\leqq 2^{1+2}$. To obtain $\pi_{1,3}$ we use Construction (I). σ is given by $(\sigma(2))(e)=1$ and $(\sigma(3))(e)=2$. $\pi_{1,3}$ is obtained (for example) by the first three rows of Table 1, and $\pi_{1,4}$, $\pi_{1,5}$, $\pi_{1,6}$ are represented by the first 4, 5, 6 rows of Table 1. These op-mappings are obtained by Construction (II.1), i.e., to satisfy the following conditions:

(i) The x_1-component of the i-th row equals the e-component of the $(i+1)$-th row.

(ii) All rows are distinct.

We can not continue this procedure to give $\pi_{1,7}$. Because two rows, i.e., two elements of $F_1(X, Y)$, which are not yet used are (2, 2, 1) and (2, 2, 2), and we can not determine the 7th row so as to satisfy (i) and (ii). This means that we are in case (II.2). As shown in Lemma 14, the x_1-component of the 6th row is equal to the e-component of the first row. We make a cyclic exchange of 6 rows for example in Table 2. Then we can add the 7th and 8th rows to satisfy (i) and (ii). In this way, we have $\pi_{1,7}$ and $\pi_{1,8}$ which are the first 7 and 8 rows of Table 2.

For $t=2$, $\pi_{2,s}$ exist for $2^{1+2}+1\leqq s\leqq 2^{1+2+4}$. To construct $\pi_{2,9}$, first obtain σ from $\pi_{1,8}$. σ is shown in Table 3 which is derived by cyclic exchange of Table 2 so that the top row is (1, 1, 1). The first 9 rows of Table 4 are constructed as follows:

(i) All components of the first row are 1.

(ii) The e-, x_1- and x_2-components from the 2nd to the 9th rows are coincident with those of σ.

(iii) The x_1x_1- and x_1x_2-components of the i-th row are equal to the x_1- and x_2-components of the $(i+1)$-th row ($i\in[2: 8]$).

(iv) The x_2x_1- and x_2x_2-components from the 2nd to the 9th rows are arbitrarily chosen. The x_1x_1- and x_1x_2-components of the 9th row are also arbitrarily chosen.

In this way we have $\pi_{2,9}$ by using Construction (I). To obtain $\pi_{2,s}$ for $s=10, 11, \ldots,$ we add new rows one by one so that the following conditions are satisfied (Construction (II.1)).

(i) The x_1x_1- and x_1x_2-components of the i-th row are equal to the x_1- and x_2-components of the $(i+1)$-th row.

(ii) All rows are distinct.

In the case when we can not continue this procedure (Case (II.2)), we make a cyclic exchange of rows and continue the procedure. In such a way, we can obtain $\pi_{2,s}$ for all $s\in[9: 2^7]$. Table 4 shows $\pi_{2,s}$ for $s\in[9: 16]$.

Table 1

	e	x_1	x_2
1	1	1	1
2	1	2	1
3	2	1	1
4	1	1	2
5	1	2	2
6	2	1	2

Table 2

	e	x_1	x_2
1	2	1	1
2	1	1	2
3	1	2	2
4	2	1	2
5	1	1	1
6	1	2	1
7	2	2	1
8	2	2	2

Table 3

	e	x_1	x_2
2	1	1	1
3	1	2	1
4	2	2	1
5	2	2	2
6	2	1	1
7	1	1	2
8	1	2	2
9	2	1	2

Table 4

	e	x_1	x_2	x_1x_1	x_1x_2	x_2x_1	x_2x_2
1	1	1	1	1	1	1	1
2	1	1	1	2	1	1	1
3	1	2	1	2	1	1	1
4	2	2	1	2	2	2	1
5	2	2	2	1	1	2	2
6	2	1	1	1	2	1	2
7	1	1	2	2	2	1	1
8	1	2	2	1	2	1	2
9	2	1	2	1	1	1	2
10	1	1	1	1	2	1	2
11	1	1	2	2	2	1	2
12	1	2	2	2	2	2	2
13	2	2	2	1	2	1	2
14	2	1	2	2	2	1	2
15	1	2	2	1	1	1	1
16	2	1	1	1	1	1	1

Next we shall see two examples of realization of 4-tuples (v, n, m, w).

Let $(v, n, m, w)=(10, 2, 2, 4)$. Since $2+4\leqq 10\leqq 2^{1+2+2^3+2^4}$, $(10, 2, 2, 4)$ is realizable by initially connected Moore automata. The unique solution of $2^{1+2+\cdots+2^{t-1}}+2\leqq 10-4+t\leqq 2^{1+2+2^2+\cdots+2^t}$ is $t=1$. Hence $\mathrm{A}(\pi_{1,7}, 3)$ realizes $(10, 2, 2, 4)$. In Fig. 1, an example of $\mathrm{A}(\pi_{1,7}, 3)$ is depicted, which is obtained by using Table 2.

Let $(v, n, m, w)=(17, 2, 2, 5)$. Since $2+5 \leqq 17 \leqq 2^{1+2+2^3+2^4+2^5}$, $(17, 2, 2, 5)$ is realizable by initially connected Moore automata. The unique solution of $2^{1+2+\cdots+2^{t-1}}+2 \leqq 17-5+t \leqq 2^{1+2+2^2+\cdots+2^t}$ is $t=2$. Hence $\mathrm{A}(\pi_{2,14}, 3)$ realizes $(17, 2, 2, 5)$. $\mathrm{A}(\pi_{2,14}, 3)$ is illustrated in Fig. 2.

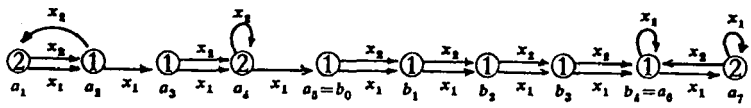


Fig. 1

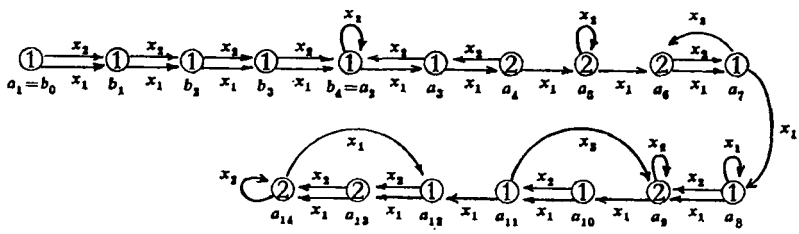


Fig. 2

Acknowledgement

The author would like to thank Professors A. Ádám and M. Ito for their careful readings and comments.

FACULTY OF SCIENCE
KYOTO SANGYO UNIVERSITY
KYOTO 603, JAPAN

References

[1] Ádám, A., On the question of description of the behavior of finite automata, Studia Sci. Math. Hungar. 13 (1978), 105—124.
[2] Ádám, A., On the complexity of codes and pre-codes assigned to finite Moore automata, Acta Cybernet., 5 (1981), 117—133.
[3] Ádám, A., Research problem 29, The connection of state number and the complexity of finite Moore automata, Period. Math. Hungar., 12 (1981), 229—230.
[4] Ádám, A., On certain partitions of finite directed graphs and of finite automata, Acta Cybernet. 6 (1984), 331—346.
[5] Ito, M. and J. Duske, On cofinal and definite automata, Acta Cybernet., 6 (1983), 181—189.

(Received Febr. 1, 1985)

Varieties and general products of top-down algebras

By Z. ÉSIK

Unrestricted, i.e. both finite and infinite general products of unoids were treated in [2]. It has been shown that the unoid varieties arising with general products are exactly those classes of unoids which have equational presentation in terms of so-called p-identities. In addition, these type independent varieties coincide with the varieties obtainable with the more special α_0-products. In other words this means that the unrestricted general product is homomorphically as general as the α_0-product. Although unoids do have certain specialities as shown in [1] and [2], using a new method, the above mentioned results have been extended to arbitrary algebras in [1]. Due to the specific nature of unoids, all type-independent varieties of unoids have been described in [2]. No similar description is attainable for the general case of algebras at present.

The aim of this paper is to give similar results for top-down algebras, a less well-known type of algebraic structures originating from tree automata theory. Top-down algebras are elsewhere called root-to-frontier algebras or ascending algebras as eg. in [3] and [4], due to a converse visualization of trees. The whole treatment will be done parallel with [1].

1. Top-down algebras and general products

Let R be a nonvoid subset of the natural numbers $N=\{1, 2, \ldots\}$. R is called a rank type and will be fixed throughout the paper. A type of rank type R is a collection $F=\bigcup(F_n| n\in N)$ so that $F_n\neq\emptyset$ if and only if $n\in R$. In the sequel every type F is supposed to belong to the fixed rank type R. A top-down F-algebra is an ordered pair $\mathfrak{A}=(A, F)$ with A a nonvoid set and a realization $f: A\to A^n$ for each operational symbol $f\in F_n$. The class of all top-down F-algebras is denoted $\mathbf{K}_F$. $\mathbf{K}_R=$ $=\bigcup(\mathbf{K}_F|F$ has rank type $R)$.

Suppose we are given two top-down F-algebras $\mathfrak{A}=(A, F)$ and $\mathfrak{B}=(B, F)$. A mapping $\varphi: A\to B$ is called a homomorphism $\mathfrak{A}\to\mathfrak{B}$ if $\varphi f pr_i=f pr_i\varphi$ for every $\in F_n$ and $i\in[n]=\{1, \ldots, n\}$, where pr_i denotes the i-th projection and functional composition is written juxtaposition from left to right. If φ is onto, $\mathfrak{B}$ is a homomorphic image of $\mathfrak{A}$. Further, $\mathfrak{A}$ is called a subalgebra of $\mathfrak{B}$ if $A\subseteq B$ and the natural inclusion is a homomorphism $\mathfrak{A}\to\mathfrak{B}$.

Let $\mathbf{K}$ be an arbitrary class of top-down algebras (of rank type R). Then $\mathscr{H}(\mathbf{K})$

and $\mathscr{S}(\mathbf{K})$ will respectively denote the class of all homomorphic images and the class of all subalgebras of top-down algebras from $\mathbf{K}$.

Now we are going to introduce general products of top-down algebras. To this, take types F and F^i $(i\in I)$ as well as top-down F^i-algebras $\mathfrak{A}_i=(A_i, F^i)$ $(i\in I)$. Let φ be a family of feedback functions $\varphi_i: \Pi(A_i|i\in I)\times F\to F^i$. The φ_i's must preserve the rank, i.e.,

$$(a, f)\varphi_i\in F^i_n$$

if $f\in F_n$, $a\in\Pi(A_i|i\in I)$. The general product of the $\mathfrak{A}_i$'s w.r.t. φ and F is that top-down F-algebra

$$\mathfrak{A} = (A, F) = \Pi(\mathfrak{A}_i|i\in I, F, \varphi)$$

satisfying

$$A = \Pi(A_i|i\in I),$$

$$afpr_j pr_i = apr_i f^i pr_j$$

where $a\in A, f\in F_n, i\in I, j\in[n]$ and $f^i=(a,f)\varphi_i$. A general product of a finite number of top-down algebras is denoted $\Pi(\mathfrak{A}_1, \ldots, \mathfrak{A}_n|\varphi, F)$.

Two restricted forms of the general product will be of particular interest. These are the α_0-product and the direct product. The general product defined above is an α_0-product if the index set I is linearly ordered, and for every $i\in I$, the feedback function φ_i assigning value in F^i to $((a_i|i\in I), f)\in\Pi(A_i|i\in I)\times F$ is independent of the a_j's with $j\geqq i$. In case of an α_0-product we shall indicate only those variables of φ_i on which it may depend. Index sets $[n]$ are supposed to have the natural ordering.

The concept of the direct product easily comes by specialization, too. A general product $\mathfrak{A}=(A, F)=\Pi(\mathfrak{A}_i|i\in I, \varphi, F)$ is a direct product if all factors $\mathfrak{A}_i$ are top-down F-algebras and $(a, f)\varphi_i=f, i\in I, f\in F, a\in A$.

Take a class $\mathbf{K}$ of top-down algebras. The operators $\mathscr{P}_g$, $\mathscr{P}_{\alpha_0}$, $\mathscr{P}$ and $\mathscr{P}_{f\alpha_0}$ are defined by the following list:

$\mathscr{P}_g(\mathbf{K})$: all general products of factors from $\mathbf{K}$,
$\mathscr{P}_{\alpha_0}(\mathbf{K})$: all α_0-products of factors from $\mathbf{K}$,
$\mathscr{P}(\mathbf{K})$: all direct products of factors from $\mathbf{K}$,
$\mathscr{P}_{f\alpha_0}(\mathbf{K})$: all α_0-product of finitely many factors from $\mathbf{K}$.

According to the universal algebraic analogy (see also the next section), classes $\mathbf{K}\subseteqq\mathbf{K}_F$ closed under the operators $\mathscr{H}$, $\mathscr{S}$ and $\mathscr{P}$ are called varieties. However, the main interest will be in type-independent varieties. By definition, a type-independent variety is a class $\mathbf{K}\subseteqq\mathbf{K}_R$ closed under the operators $\mathscr{H}$, $\mathscr{S}$ and $\mathscr{P}_g$.

2. Varieties of top-down algebras

Top-down algebras of rank type $R=\{1\}$ will be called unoids. Since in a unoid (A, F) every operation is a function $f: A\to A$, unoids are ordinary algebras.

Let $\mathfrak{A}=(A, F)$ be a top-down F-algebra. There is a simple way to associate a unoid $\mathfrak{A}^u=(A, F^u)$ with $\mathfrak{A}$: put $a(f,i)=afpr_i$ for every $a\in A$ and $(f, i)\in F^u$. Here the unary type F^u consists of all pairs (f, i) with $f\in F_n$ and $i\in[n]$. It is obvious that every homomorphism from a top-down F-algebra $\mathfrak{A}$ into a top-down F-algebra

$\mathfrak{B}$ becomes a homomorphism $\mathfrak{A}^u \to \mathfrak{B}^u$, and the resulting functor is an isomorphism of the category of all top-down F-algebras onto the category of all F^u-unoids. Varieties are preserved under this transition, if $\mathbf{V} \subseteq \mathbf{K}_F$ is a variety, then so is $\mathbf{V}^u = \{\mathfrak{A}^u | \mathfrak{A} \in \mathbf{V}\}$, and conversely. Anyway, this simple transition allows us to adapt well-known concepts and facts from universal algebra to our top-down algebras, e.g., for any class $\mathbf{K} \subseteq \mathbf{K}_F$, $\mathscr{HSP}(\mathbf{K})$ is least variety containing $\mathbf{K}$.

The concept of an identity also extends to our case in an obvious way. An F-identity is either a formal equation $xp = xq$ or $xp = yq$, where x and y are different variables, p and q are words over the alphabet F^u, i.e., $p, q \in (F^u)^*$. Expressions zp with $z \in \{x, y\}$ and $p \in (F^u)^*$ are called polynomial symbols. The number of letters appearing in zp is called the length of zp and is denoted $|zp|$. The set $\mathrm{pre}(zp)$ is defined by $\mathrm{pre}(zp) = \{zq \,|\, |zq| < |zp|, \exists r\ qr = p\}$. An F-identity is satisfied by an F-algebra $\mathfrak{A}$ if it is satisfied by the unoid $\mathfrak{A}^u$ in the ordinary sense. In this case we also say the F-identity holds in $\mathfrak{A}$.

For a class $\mathbf{K} \subseteq \mathbf{K}_F$, $\mathrm{Id}(\mathbf{K})$ denotes the set of all identities satisfied by every $\mathfrak{A} \in \mathbf{K}$. Further, if Σ is a set of F-identities, then $\mathrm{Mod}(\Sigma)$ is the class of all top-down F-algebras satisfying every F-identity in Σ. We write $\Sigma \models \Delta$ to mean that $\mathrm{Mod}(\Sigma) \subseteq \mathrm{Mod}(\Delta)$.

With these concepts in mind one can easily reformulate Birkhoff's Theorem for top-down algebras. A class $\mathbf{K} \subseteq \mathbf{K}_F$ is a variety if and only if $\mathbf{K} = \mathrm{Mod}(\Sigma)$ for a set of F-identities Σ. Σ can be chosen $\mathrm{Id}(\mathbf{K})$. Consequently, $\mathscr{HSP}(\mathbf{K}) = \mathrm{Mod}(\mathrm{Id}(\mathbf{K}))$ for any class $\mathbf{K}$.

A crucial point in the universal algebraic proof of Birkhoff's Theorem is the existence of all free algebras in a variety. Free algebras exist in varieties of top-down algebras, too. If $\mathbf{V} \subseteq \mathbf{K}_F$ is a variety of top-down algebras then a free algebra $\mathfrak{A} = (A, F) \in \mathbf{V}$ with free generator $a \in A$ has the following property. An F-identity $xp = xq$ holds in $\mathbf{V}$ if and only if $ap = aq$. Similarly, if $\mathfrak{A} \in \mathbf{V}$ is freely generated by $a_1, a_2 \in A$, an F-identity $xp = yq$ belongs to $\mathrm{Id}(\mathbf{V})$ if and only if $a_1 p = a_2 q$.

3. Type-independent varieties

In this section we are going to develop a theory of type-independent varieties of top-down algebras similar to the theory of varieties exhibited in the previous one. To start with notice

Statement 1. For every class $\mathbf{K}$, $\mathscr{HSP}_g(\mathbf{K})$ is the least type-independent variety containing $\mathbf{K}$.

To show that type-independent equational classes also have equational characterizations we now introduce the notion of a p-identity. There are 3 types of p-identities, namely

(i) $(u, v) = (u, w)$,

(ii) $(u, z_1, v) = (u, z_2, w)$,

(iii) $v = w$

where u, v, w are possibly void words in $\{(n, i) | n \in R, i \in [n]\}^*$, and $z_1, z_2 \in \{(n, i) | n \in R, i \in [n]\}$. In more detail, say $u = (l_1, i_1) \ldots (l_r, i_r)$, $v = (m_1, j_1) \ldots (m_s, j_s)$, $w = (n_1, k_1) \ldots$

$\ldots(n_t, k_t)$, $z_1=(d,i)$ and $z_2=(d,j)$. It is required that $i\neq j$. Given a type F, each of these p-identities induces a set of F-identities. These are given by the formulae below:

(i′) $xpq_1 = xpq_2$,

(ii′) $xp(f,i)q_1 = xp(f,j)q_2$,

(iii′) $xq_1 = yq_2$

where $p=(f_1,i_1)\ldots(f_r,i_r)$, $q_1=(g_1,j_1)\ldots(g_s,j_s)$, $q_2=(h_1,k_1)\ldots(h_t,k_t)$, further, $f_1\in F_{l_1},\ldots,f_r\in F_{l_r}$, $g_1\in F_{m_1},\ldots,g_s\in F_{m_s}$, $h_1\in F_{n_1},\ldots,h_t\in F_{n_t}$, and finally, $f\in F_d$. A p-identity is said to be satisfied by a top-down F-algebra $\mathfrak{A}$ if all its induced F-identities are satisfied by $\mathfrak{A}$. Alternatively, this is expressed by saying the p-identity holds in $\mathfrak{A}$.

Let Ω be a set of p-identities. The set of F-identities induced by p-identities from Ω is denoted Ω_F. Ω^* denotes the class of all top-down algebras satisfying every member of Ω. Further, if $\mathbf{K}$ is an arbitrary class of top-down algebras, $\mathbf{K}^*$ is the set of all p-identities which hold in every $\mathfrak{A}\in\mathbf{K}$.

The following proposition easily comes from the definitions.

Statement 2. $(\mathscr{HSP}_g(\mathbf{K}))^*=\mathbf{K}^*$ holds for every class $\mathbf{K}$.

The clue in our treatment is

Lemma 1. If $\mathbf{K}$ is a type-independent variety then $\mathrm{Id}\,\mathbf{K}_F^*|=\mathrm{Id}\,(\mathbf{K}\cap\mathbf{K}_F)$.

Proof. Assume to the contrary there exists an F-identity in $\mathrm{Id}\,(\mathbf{K}\cap\mathbf{K}_F)$ which is not a consequence of $\mathbf{K}_F^*$. Among these there is one having minimum weight. The weight of on F-identity $xp=yq$ is defined $\mathrm{card}\,(\mathrm{pre}\,(xp)\cup\mathrm{pre}\,(xq))$. Similarly, the weight of $xp=yq$ is just $\mathrm{card}\,(\mathrm{pre}\,(xp)\cup\mathrm{pre}\,(yq))$. We shall restrict ourselves to the case this minimum weight F-identity is $xp=xq$. The other case can be handled likewise.

Take a free algebra $\mathfrak{A}=(A,F)$ in the variety $\mathbf{K}\cap\mathbf{K}_F$ with free generator a. We are going to show that whenever $xr, xs\in\mathrm{pre}\,(xp)\cup\mathrm{pre}\,(xq)$ and $ar=as$, then xr and xs coincide. Thus, suppose $xr, xs\in\mathrm{pre}\,(xp)\cup\mathrm{pre}\,(xq)$ and $ar=as$. We may choose xr and xs so that $|xr|\leqq|xt|\leqq|xs|$ provided that $xt\in\mathrm{pre}\,(xp)\cup\mathrm{pre}\,(xq)$ and $ar=at\,(=as)$. Let us substitute xr for xs if $xs\in\mathrm{pre}\,(xp)$, and apply the same substitution for xq. Denote the resulting polynomial symbols by $x\bar{p}$ and $x\bar{q}$, respectively. If xr is different from xs then both F-identities $x\bar{p}=x\bar{q}$ and $xr=xs$ have weight strictly less than that of $xp=xq$. On the other hand, $ar=as$ and $a\bar{p}=ap=aq=a\bar{q}$ yield $xr=xs$, $x\bar{p}=x\bar{q}\in\mathrm{Id}\,(\mathbf{K}\cap\mathbf{K}_F)$. By the choice of $xp=xq$ we have $\mathbf{K}_F^*|=\{xr=xs,\ x\bar{p}=x\bar{q}\}$, while $\{xr=xs,\ x\bar{p}=x\bar{q}\}|=\{xp=xq\}$ follows via the construction. Therefore, $\mathbf{K}_F^*|=\{xp=xq\}$. This contradiction arose from the assumption xr is different from xs, hence, xr and xs coincide.

Write xp and xq in more detail as

$$xp = x(f_1,i_1)\ldots(f_r,i_r)(g_1,j_1)\ldots(g_s,j_s),$$

$$xq = x(f_1,i_1)\ldots(f_r,i_r)(h_1,k_1)\ldots(h_t,k_t),$$

where $f_1\in F_{l_1},\ldots,f_r\in F_{l_r}$, $g_1\in F_{m_1},\ldots,g_s\in F_{m_s}$, $h_1\in F_{n_1},\ldots,h_t\in F_{n_t}$ and $(g_1,j_1)\neq$

$\neq(h_1, k_1)$ if $s, t>0$. First suppose that $g_1\neq h_1$ if $s, t>0$. Since $\mathbf{K}_F^*\models\{xp=xq\}$, $xp=xq\notin\mathbf{K}_F^*$. Consequently, the p-identity

$$((l_1, i_1) \dots (l_r, i_r), (m_1, j_1) \dots (m_s\ j_s)) = ((l_1, i_1) \dots (l_r, i_r), (n_1, k_1) \dots (n_t, k_t))$$

is not in $\mathbf{K}^*$. This means that there exist a top-down F'-algebra $\mathfrak{B}=(B, F')\in\mathbf{K}$, operational symbols $f_1'\in F_{i_1}', \dots, f_r'\in F_{l_r}'$, $g_1'\in F_{m_1}', \dots, g_s'\in F_{m_s}'$, $h_1'\in F_{n_1}', \dots, h_t'\in F_{n_t}'$, and an element $b\in B$ with

$$bp' = b(f_1', i_1) \dots (f_r', i_r)(g_1', j_1) \dots (g_s', j_s) \neq$$
$$\neq b(f_1', i_1) \dots (f_r', i_r)(h_1', k_1) \dots (h_t', k_t) = bq'.$$

Now define an α_0-product $\mathfrak{Q}=\Pi(\mathfrak{A}, \mathfrak{B}|\varphi, F)$ so that $\varphi_1\colon F\to F$ is the identity function and $\varphi_2\colon A\times F\to F'$ is any function with

$$(a(f_1, i_1) \dots (f_u, i_u), f_{u+1})\varphi_2 = f_{u+1}', \quad u = 0, \dots, r-1,$$
$$(a(f_1, i_1) \dots (f_r, i_r)(g_1, j_1) \dots (g_u, j_u), g_{u+1})\varphi_2 = g_{u+1}',$$
$$u = 0, \dots, s-1,$$
$$(a(f_1, i_1) \dots (f_r, i_r)(h_1, k_1) \dots (h_u, k_u), h_{u+1})\varphi_2 = h_{u+1}',$$
$$u = 0, \dots, t-1.$$

The first part of the proof garantees the existence of such an α_0-product. It is easy to check that

$$(a, b)ppr_2 = bp' \neq bq' = (a, b)qpr_2,$$

thus $xp=xq\notin \mathrm{Id}(\{\mathscr{L}\})$. Since $\mathscr{L}\in\mathbf{K}\cap\mathbf{K}_F$, this gives a contradiction.

The second case, i.e. when $s, t>0$ and $g_1=h_1$ yields a similar contradiction just take the p-identity

$$((l_1, i_1) \dots (l_r, i_r), (m_1, j_1), (m_2, j_2) \dots (m_s, j_s)) =$$
$$= ((l_1, i_1) \dots (l_r, j_r), (n_1, k_1), (n_2, k_2) \dots (n_t, k_t)).$$

Remark. Since only α_0-products were used in the previous proof, the statement of Lemma 1 holds even if closure under $\mathscr{P}_{\alpha_0}$ is supposed instead of closure under $\mathscr{P}_g$. Furthermore, closure under $\mathscr{H}$ and $\mathscr{S}$ is not strictly required.

Lemma 2. $\mathrm{Id}(\mathscr{P}_{\alpha_0}(\mathbf{K})\cap\mathbf{K}_F)=\mathrm{Id}(\mathscr{P}_{f\alpha_0}(\mathbf{K})\cap\mathbf{K}_F)$ holds for every class $\mathbf{K}$ and type F.

Proof. This statement has been proved for ordinary algebras in [1]. The same idea applies here. However, it should be noted that [4] also contains the proof.

We are ready to prove the main result:

Theorem. For any class $\mathbf{K}$ of top-down algebras, $\mathbf{K}^{**}=\mathscr{H}\mathscr{S}\mathscr{P}_g(\mathbf{K})=\mathscr{H}\mathscr{S}\mathscr{P}_{\alpha_0}(\mathbf{K})=$ $=\mathscr{H}\mathscr{S}\mathscr{P}\mathscr{P}_{f\alpha_0}(\mathbf{K})$.

Proof. $\mathbf{K}^{**}\supseteq\mathscr{H}\mathscr{S}\mathscr{P}_g(\mathbf{K})$ is valid by Statement 2. Inclusions $\mathscr{H}\mathscr{S}\mathscr{P}_g(\mathbf{K})\supseteq$ $\supseteq\mathscr{H}\mathscr{S}\mathscr{P}_{\alpha_0}(\mathbf{K})\supseteq\mathscr{H}\mathscr{S}\mathscr{P}\mathscr{P}_{f\alpha_0}(\mathbf{K})$ are trivial. $\mathscr{H}\mathscr{S}\mathscr{P}_{\alpha_0}(\mathbf{K})\supseteq\mathbf{K}^{**}$ follows by Lemma 1 and the Remark. Finally, $\mathscr{H}\mathscr{S}\mathscr{P}\mathscr{P}_{f\alpha_0}(\mathbf{K})\supseteq\mathscr{H}\mathscr{S}\mathscr{P}_{\alpha_0}(\mathbf{K})$ is valid by Lemma 2.

Corollary. The following three statements are equivalent for every class $\mathbf{K}$:

(i) $\mathbf{K}$ is a type independent equational class,
(ii) $\mathbf{K}=\Omega^*$ for a set Ω of p-identities,
(iii) $\mathbf{K}=\mathbf{K}^{**}$.

Note. Equations $\mathscr{HSP}_g(\mathbf{K})=\mathscr{HSP}_{\alpha_0}(\mathbf{K})=\mathscr{HSPP}_{f\alpha_0}(\mathbf{K})$ have already been established in [4].

A. JÓZSEF UNIVERSITY
BOLYAI INSTITUTE
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] Z. ÉSIK, On identities preserved by general products of algebras, Acta Cybernet., 6 (1983), 285—289.
[2] Z. ÉSIK and F. GÉCSEG, General products and equational classes of automata, Acta Cybernet., 6 (1983), 281—284.
[3] F. GÉCSEG, On a representation of deterministic uniform root-to-frontier tree transformations, Acta Cybernet., 6 (1983), 173—180.
[4] F. GÉCSEG, Metric equivalence of tree automata, Acta Sci. Math., 48 (1985), 163—171.

(Received Febr. 6, 1985)

On products of automata with identity

By Z. ÉSIK and J. VIRÁGH

In spite of the fascinating Krohn—Rhodes theory the homomorphically complete classes of automata have not yet been satisfactorily characterized for the α_0-product. Recently there has been keen activity in finding nice homomorphically complete classes. Continuing the work which was begun by N. V. Evtusenko [6], P. Dömösi gave a very interesting homomorphically complete class for the α_0-product consisting of automata having 3 input signs (cf. [3]). His idea was to use not only permutation automata for the homomorphic realization of permutation automata. He applied a technique combining shiftregisters with permutation automata, and in a sense his use of shiftregisters originates in [4]. It was apparent for us that Dömösi did not completely exploit the advantages of this method. The present paper is a collection of a few remarks immediately obtainable just by simple generalization.

The basic idea behind the use of shiftregisters is this. Let a part of the product automaton work in an absolutely free way by sections, and if enough information has been accumulated try to have this information govern the next move simulating the behaviour of the automaton to be realized homomorphically. Not surprisingly this has something to do with generalized products, i.e. products allowing an input sign to be coded with an input word of arbitrary length. Namely, this shiftregister technique can be used for converting generalized products to ordinary products. Unfortunately this conversion can not always be carried out. But the presence of input signs inducing the identity mapping on the state set does make the conversion possible under wide circumstances.

1. Preliminaries

We shall be using standard automata theoretic notions. An automaton is meant a system $\mathbf{A}=(A, X, \delta)$, where A and X are finite nonvoid sets, the state set and the input alphabet, and the transition function δ maps $A\times X$ into A. Denoting by X^* the free semigroup with identity λ generated by X, the transition function extends to a map $A\times X^*\to A$ as usual. Given a word $p\in X^*$, the length of p is denoted $|p|$. Every word $p\in X^*$ induces a translation $t_p^{\mathbf{A}} : A\to A$ of the state set: $t_p^{\mathbf{A}}(a)=\delta(a, p)$ for all $a\in A$. If no confusion may arise, we write t_p instead of $t_p^{\mathbf{A}}$. All translations t_p, $p\in X^*$, form a semigroup with respect to function composition. This semigroup $S(\mathbf{A})$ is called the characteristic semigroup of $\mathbf{A}$.

For every automaton $\mathbf{A}=(A, X, \delta)$, we define the automata $\mathbf{A}^\lambda$ and $\mathbf{A}^*$ as

follows: $\mathbf{A}^\lambda=(A, \{t_\lambda^{\mathbf{A}}, t_x^{\mathbf{A}}|x\in X\}, \delta^\lambda)$, $\mathbf{A}^*=(A, S(\mathbf{A}), \delta^*)$, where $\delta^\lambda(a, t_\lambda^{\mathbf{A}})=t_\lambda^{\mathbf{A}}(a)=a$, $\delta^\lambda(a, t_x^{\mathbf{A}})=t_x^{\mathbf{A}}(a)$, and $\delta^*(a, t_p^{\mathbf{A}})=t_p^{\mathbf{A}}(a)$ for any $a\in A$, $x\in X$, $p\in X^*$. Notice that $S(\mathbf{A})=S(\mathbf{A}^\lambda)=S(\mathbf{A}^*)$. For a class $\mathscr{K}$ of automata put

$$\mathscr{K}^\lambda = \{\mathbf{A}^\lambda|\mathbf{A}\in\mathscr{K}\},$$

$$\mathscr{K}^* = \{\mathbf{A}^*|\mathbf{A}\in\mathscr{K}\}.$$

Let $\mathbf{A}=(A, X, \delta)$ and $\mathbf{B}=(B, Y, \delta')$ be two automata. $\mathbf{A}$ is called an X subautomaton of $\mathbf{B}$, if $A\subseteq B$, $X\subseteq Y$, and δ is the restriction of δ' to $A\times X$. If $X=Y$, we speak about a subautomaton. Take two mappings $h_1: A\to B$ and $h_2: X\to Y$. This pair of functions is said to be an X-homomorphism $\mathbf{A}\to\mathbf{B}$ if $h_1(\delta(a, x))=\delta'(h_1(a), h_2(x))$ for every $a\in A$, $x\in X$. If in addition both h_1 and h_2 are bijective, we call the pair (h_1, h_2) an X-isomorphism, and $\mathbf{A}$ X-isomorphic to $\mathbf{B}$. Letting $X=Y$ and h_2 the identity map $X\to Y$, h_1 becomes a homomorphism $\mathbf{A}\to\mathbf{B}$. $\mathbf{B}=(B, X, \delta')$ is a homomorphic image of $\mathbf{A}$ if there is a surjective homomorphism $\mathbf{A}\to\mathbf{B}$. Bijective homomorphisms are called isomorphisms.

Take a class $\mathscr{K}$ of automata. Then $\mathbf{S}(\mathscr{K})$, $\mathbf{H}(\mathscr{K})$ and $\mathbf{I}(\mathscr{K})$ will respectively denote the classes of all subautomata, homomorphic images and isomorphic images of automata from $\mathscr{K}$.

Now we recall the concept of general products of automata. Let $\mathbf{A}_j=(A_j, X_j, \delta_j)$, $j\in[n]=\{1, \ldots, n\}$, $n\geqq 0$ be arbitrary automata and take a system of so called feedback functions $\varphi_j: A_1\times\ldots\times A_n\times X\to X_j$, $j\in[n]$, where X is any alphabet. The automaton $\mathbf{A}=(A_1\times\ldots\times A_n, X, \delta)$ will be called the general product (g-product, for short) of automata $\mathbf{A}_j$ with respect to φ and X, provided that

$$\delta((a_1, \ldots, a_n), x) = (\delta_1(a_1, x_1), \ldots, \delta_n(a_n, x_n)),$$

$$x_j = \varphi_j(a_1, \ldots, a_n, x)$$

for every $a_1\in A_1, \ldots, a_n\in A_n$, $x\in X$ and $j\in[n]$. We use the notation $\mathbf{A}_1\times\ldots\times\mathbf{A}_n(\varphi, X)$ for general products. If all the $\mathbf{A}_j$'s coincide, we speak about a power.

Take the general product above, and let $i\geqq 0$ be an arbitrary integer. If none of the feedback functions φ_j depends on the state variables a_k having indices $k>j+i-1$, the g-product is called an α_i-product. In case of an α_i-product we shall indicate only those variables of a feedback function on which it may depend.

We shall make use of an interesting generalization of g-products. Take the automata $\mathbf{A}_j$ as in the definition of a g-product but now let $\varphi_j: A_1\times\ldots\times A_n\times X\to X_j^*$, $j\in[n]$. The g^*-product $\mathbf{A}_1\times\ldots\times\mathbf{A}_n(X, \varphi)$ is defined on exact analogy of the g-product with the exception that

$$\delta(a_1, \ldots, a_n, x) = (\delta_1(a_1, p_1), \ldots, \delta_n(a_n, p_n)),$$

where $p_j=\varphi_j(a_1, \ldots, a_n, x)$, $j\in[n]$. Allowing only words of length not exceeding 1 in the ranges of the feedback functions, we get the notion of a g^λ-product, or general λ-product. Note that g-products are special g^λ-products, and g^λ-products are special cases of the g^*-product. The concept of an α_i^*-product or that of an α_i^λ-product is derived in the same way as α_i-products were obtained.

Take a class $\mathscr{K}$ of automata. We put

$\mathbf{P}_g(\mathscr{K})$: all g-products of automata from $\mathscr{K}$,

$\mathbf{P}_{\alpha_i}(\mathscr{K})$: all α_i-products of automata from $\mathscr{K}$,
$\mathbf{P}^*_g(\mathscr{K})$: all g^*-products of automata from $\mathscr{K}$,
$\mathbf{P}^*_{\alpha_i}(\mathscr{K})$: all α_i^*-products of automata from $\mathscr{K}$,
$\mathbf{P}^\lambda_g(\mathscr{K})$: all g^λ-products of automata from $\mathscr{K}$,
$\mathbf{P}^\lambda_{\alpha_i}(\mathscr{K})$: all α^i-products of automata from $\mathscr{K}$.

Observe that the following are identities:

$$P^*_g(\mathscr{K}) = P_g(\mathscr{K}^*),\ P^*_{\alpha_i}(K) = P_{\alpha_i}(\mathscr{K}^*),$$
$$P^\lambda_g(\mathscr{K}) = P_g(\mathscr{K}^\lambda),\ P^\lambda_{\alpha_i}(K) = P_{\alpha_i}(\mathscr{K}^\lambda).$$

Our principal interest will be in operators **HSP** where **P** is any of the product operators above. We shall give a sufficient condition for having $\mathbf{HSP}^*_{\alpha_0}(\mathscr{K}) = \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$, as well as a necessary and sufficient condition assuring $\mathbf{HSP}^*_{\alpha_1}(\mathscr{K}) = \mathbf{HSP}^\lambda_{\alpha_1}(\mathscr{K})$. As regards α_i-products with $i \geqq 2$, we show that $\mathbf{HSP}^\lambda_{\alpha_i}(\mathscr{K}) = \mathbf{HSP}^*_g(\mathscr{K})$ is identically valid. These are the main results. In addition, we shall discuss homomorphically complete classes. Recall that a class $\mathscr{K}$ is homomorphically complete for the g-product if $\mathbf{HSP}_g(\mathscr{K})$ is the class of all automata. Isomorphic completeness and homomorphic completeness with respect to other types of the product are similarly defined. We end the paper by presenting a class of automata which is homomorphically complete for the α_0-product and contains automata having only 2 input signs.

The concept of g-products was introduced by V. M. Gluskov in [10]. The hierarchy of α_i-products is due to F. Gécseg [8]. The α_0-product was called loop-free product or R-product earlier. Or even, the formation of α_0-products is equivalent to the iterated quasi-superposition. Generalized products appear in F. Gécseg [7]. Some elementary properties of the products will be used in the sequel without any reference.

We are indebted to Prof. F. Gécseg for inspiring conversations. His new book [9] is an excellent summary of recent results on products of automata.

2. Homomorphic realization

The reason for introducing the α_i-products was to decrease the complexity of the general product. On the other hand, it made possible the investigation of deeper structural properties of automata and, at the same time, gave a framework for achieving deep results. The cruical example is the Krohn—Rhodes theory. F. Gécseg observed how to translate this theory into the scope of α_0^*-products. His achievements will be summarized in Theorem 1. In this theorem, as well as throughout the paper, $\mathbf{A}_0$ denotes the two-state reset automaton ([2], $\{x, y\}, \delta_0$), $\delta_0(1, x) = \delta_0(2, x) = 1$, $\delta_0(1, y) = \delta_0(2, y) = 2$. The automaton $\mathbf{A}_0^\lambda$ can be identified with ([2], $\{x_0, x, y\}, \delta_0'$), where δ_0' coincides with δ_0 on $[2] \times \{x, y\}$, and x_0 induces the identity.

Theorem 1. A class $\mathscr{K}$ of automata is homomorphically complete for the α_0^*-product if and only if the following are valid:

(i) There is an automaton in $\mathscr{K}$ whose characteristic semigroup contains a subsemigroup isomorphic to $S(\mathbf{A}_0)$.

(ii) For every finite simple group G, there exists an automaton $\mathbf{A}\in\mathscr{K}$ such that G is a homomorphic image of a subgroup of $S(\mathbf{A})$.

Consequently, there exists no minimal homomorphically complete class of automata for the α_0^*-product.

Combining the proof with the Krohn—Rhodes theory one gets:

Corollary 1. Let $\mathscr{K}$ be a class satisfying (i) above, and take an automaton $\mathbf{A}$. Then $\mathbf{A}\in\mathbf{HSP}^*_{\alpha_0}(\mathscr{K})$ if and only if whenever a simple group G is a homomorphic image of a subgroup of $S(\mathbf{A})$, there is an automaton $\mathbf{B}\in\mathscr{K}$, for which a subgroup of $S(\mathbf{B})$ can be mapped homomorphically onto G. A part of this holds for any class $\mathscr{K}$. Namely, whenever a simple group G is a homomorphic image of a subgroup of $S(\mathbf{A})$ and $\mathbf{A}\in\mathbf{HSP}^*_{\alpha_0}(\mathscr{K})$, then a subgroup of $S(\mathbf{B})$ can be mapped homomorphically onto G for an automaton $B\in\mathscr{K}$.

We think the above theorem clearly justifies the importance of generalized products. Our present purpose is to show that generalized products can be replaced by λ-products in most cases as far as homomorphic realization is concerned with. Theorem 1 will be our starting point for α_0^λ-products, and we shall make an attempt to combine it with a technique used by P. Dömösi in [3].

First of all we need a few concepts. Automata $\mathbf{C}_n=(\{a_1, \ldots, a_n\}, \{x\}, \delta)$ satisfying $\delta(a_i, x)=a_{i+1}$ $(i=1, \ldots, n-1)$, $\delta(a_n, x)=a_1$ will be called counters. Counters of one state are said to be trivial. An automaton $\mathbf{A}=(A, X, \delta)$ is called counter-free if and only if, whenever a counter $\mathbf{C}$ is an X-subautomaton of $\mathbf{A}$, it follows that $\mathbf{C}$ is trivial. In other words this means that $\delta(a_1, x)=a_2, \ldots, \delta(a_{n-1}, x)=a_n$, $\delta(a_n, x)=a_1$ implies $n=1$ for all $x\in X$ and different states $a_1, \ldots, a_n\in\mathbf{A}$. A class $\mathscr{K}$ of automata is counter-free if every $\mathbf{A}\in\mathscr{K}$ is counter-free.

Besides counters we shall be using shiftregisters. Let X be an alphabet. A shift-register over X of length $n\geqq 1$ is an automaton (X^n, X, δ) with transitions $\delta(x_1 \ldots x_n, x)=x_2 \ldots x_n x$, $x_1 \ldots x_n\in X^n$, $x\in X$.

Let X and Y be arbitrary alphabets and take a mapping $\tau: X^n\to Y^n$, $n\geqq 1$. Following the ideas of P. Dömösi we put $R_\tau=\{(p, q)\in X^*\times Y^* \mid 1\leqq|p|, |q|\leqq n, |p|+|q|=n+1\}$ and define the automaton $\mathbf{R}_\tau=(R_\tau, X, \delta_\tau)$ as follows:

$$\delta_\tau((p, yq), x)=\begin{cases}(px, q) & \text{if } |p|\neq n,\\ (x, \tau(p)) & \text{if } |p|=n,\end{cases}$$

where $x\in X$, $(p, yq)\in R_\tau$ with $y\in Y$.

Lemma 1. Let $\mathbf{C}_n$ be an n-state counter. Then $\mathbf{R}_\tau\in\mathbf{HSP}_{\alpha_0}(\{\mathbf{C}_n, \mathbf{A}_0\})$.

Proof. The proof is a slight modification of Dömösi's construction.

Let $\mathbf{A}_1=\mathbf{C}_n=([n], \{x_0\}, \delta_1)$ be an n-state counter, $\mathbf{A}_2=(X^n, X, \delta_2)$ a shiftregister, and set $\mathbf{A}_3=(Y^n, \overline{Y}^n\cup Y, \delta_3)$, where $\overline{Y}=\{\bar{y} \mid y\in Y\}$ and

$$\delta_3(y_1 \ldots y_n, y)=y_2 \ldots y_n y,$$

$$\delta_3(y_1 \ldots y_n, \bar{z}_1 \ldots \bar{z}_n)=z_1 \ldots z_n,$$

all $y_1 \dots y_n \in Y^n$, $\bar{z}_1 \dots \bar{z}_n \in \bar{Y}^n$, $y \in Y$. Form the α_0-product $\mathbf{A} = \mathbf{A}_1 \times \mathbf{A}_2 \times \mathbf{A}_3(\varphi, X)$ with

$$\varphi_1(x) = x_0,$$

$$\varphi_2(i, x) = x,$$

$$\varphi_3(i, x_1 \dots x_n, x) = \begin{cases} \overline{\tau(x_1 \dots x_n)} & \text{if } i = n \\ \text{arbitrary } y \in Y & \text{if } i \neq n, \end{cases}$$

$x \in X$, $i \in [n]$, $x_1 \dots x_n \in X^n$.

It is easy to check that the assignment $(i, x_1 \dots x_n, y_1 \dots y_n) \to (x_{n-i+1} \dots x_n, y_1 \dots y_{n-i+1})$ gives a homomorphism $\mathbf{A} \to \mathbf{R}_\tau$. On the other hand, both $\mathbf{A}_2$ and $\mathbf{A}_3$ are definite automata of degree n. Recall that an automaton (B, Z, δ) is called definite of degree n, if and only if $\delta(b, w) = \delta(c, w)$ holds for every $b, c \in B$ and $w \in Z^n$. Thus, $\mathbf{A}_2, \mathbf{A}_3 \in \mathbf{ISP}_{\alpha_0}(\{\mathbf{A}_0\})$ by a result of B. Imreh (cf. [11]). (Note that also the Krohn—Rhodes theorem helps in establishing $\mathbf{A}_2, \mathbf{A}_3 \in \mathbf{HSP}_{\alpha_0}(\{\mathbf{A}_0^\lambda\})$ what would be enough for our purposes in this section.) Since $\mathbf{A}_2, \mathbf{A}_3 \in \mathbf{ISP}_{\alpha_0}(\{\mathbf{A}_0\})$ and $\mathbf{R}_\tau \in \in \mathbf{HSP}_{\alpha_0}(\{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_2\})$, it follows that $\mathbf{R}_\tau \in \mathbf{HSP}_{\alpha_0}(\{\mathbf{C}_n, \mathbf{A}_0\})$.

Lemma 2. If $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$ contains a nontrivial counter then $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$ contains an infinite number of counters of different lengths.

Proof. This statement was proved in [3].

The following theorem will bear fundamental importance in our discussions.

Theorem 2. Suppose that $\mathscr{K}$ is not counter-free and $\mathbf{A}_0 \in \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$. Then $\mathscr{K}^* \subseteqq \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$.

Proof. Take an automaton $\mathbf{A} = (A, X, \delta) \in \mathscr{K}$. Then $\mathbf{A}^\lambda \in \mathbf{P}^\lambda_{\alpha_0}(\mathscr{K})$, whence we may assume that there is a sign $x_0 \in X$ inducing the identity mapping $A \to A$. We are going to show that $\mathbf{A}^* = (A, S(\mathbf{A}), \delta^*) \in \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$. Let $S(\mathbf{A}) = \{t^{\mathbf{A}}_{p_1}, \dots, t^{\mathbf{A}}_{p_k}\} = Y$, where $p_1, \dots p_k$ are words in X^*. Since x_0 induces the identity mapping $A \to A$, the words p_i can be picked out so that $|p_1| = \dots = |p_k| = n$. Or even, the previous lemma makes possible to choose n in such a manner that an n-state counter is in $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$. Obviously, there exists a mapping $\tau: Y^n \to X^n$ satisfying the equation $t^{\mathbf{A}^*}_w = t^{\mathbf{A}}_{\tau(w)}$ for every $w \in Y^n$. We form an α_0-product of R_τ and $\mathbf{A}$ and show that $\mathbf{A}^*$ is a homomorphic image of this product. Since $\mathbf{R}_\tau \in \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$, this yields $\mathbf{A}^* \in \mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$.

Take the α_0-product $\mathbf{R}_\tau \times \mathbf{A}(\varphi, Y)$ with $\varphi_1(y) = y$ and $\varphi_2((p, xq), y) = x$, and define the mapping $h: R_\tau \times A \to A$ by $h((p, q), a) = \delta^*(\delta(a, q), p)$. Then h is a homomorphism of the product onto $\mathbf{A}^*$, ending the proof of Theorem 2.

Theorem 3. Suppose that a class $\mathscr{K}$ of automata is not counter-free and the reset automaton $\mathbf{A}_0$ is in $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$. Then $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K}) = \mathbf{HSP}^*_{\alpha_0}(\mathscr{K})$. Further, an automaton $\mathbf{A}$ is in $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$ if and only if, whenever a simple group G is a homomorphic image of a subgroup of $S(\mathbf{A})$, then G is a homomorphic image of a subgroup of $S(\mathbf{B})$ for an automaton $\mathbf{B} \in \mathscr{K}$.

Proof. The inclusion $\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K}) \subseteqq \mathbf{HSP}^*_{\alpha_0}(\mathscr{K})$ is obviously valid. Con-

[1] For a word $y_1 \dots y_n \in Y^n$, $\overline{y_1 \dots y_n} = \bar{y}_1 \dots \bar{y}_n$.

versely, $\mathbf{HSP}^*_{\alpha_0}(\mathscr{K})=\mathbf{HSP}_{\alpha_0}(\mathscr{K}^*)\subseteq\mathbf{HSP}_{\alpha_0}\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})=\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$ follows by Theorem 2. The second statement is a consequence of the first one and of Corollary 1.

Corollary 2. A class $\mathscr{K}$ of automata is homomorphically complete for the α_0^λ-product if and only if the following conditions hold:

(i) $\mathscr{K}$ is not counter-free,
(ii) $\mathbf{A}_0\in\mathbf{HSP}^\lambda_{\alpha_0}(\mathscr{K})$,
(iii) for every finite simple group G, there exists an automaton $\mathbf{A}\in\mathscr{K}$ such that G is a homomorphic image of a subgroup of $S(\mathbf{A})$.

Proof. The sufficiency follows by Theorem 3. The necessity of condition (ii) is trivial, while the necessity of (iii) comes from Theorem 3. P. Dömösi proved in [2] that no counter-free class can be homomorphically complete for the α_0-product. The reason is that only the trivial counters are in $\mathbf{HSP}_{\alpha_0}(\mathscr{K})$ if $\mathscr{K}$ is counter-free.

Example 1. For every $n\geqq 1$, let $\mathbf{A}_n$ be an automaton whose characteristic semigroup is isomorphic to the symmetric group S_n of all permutations $[n]\to[n]$. The class consisting of $\mathbf{A}_0$ and these automata $\mathbf{A}_n$ $(n\geqq 1)$ is homomorphically complete for the α_0^λ-product. Consequently, $\mathscr{K}^\lambda$ is homomorphically complete for the α_0-product. Since S_n can be generated by 2 permutations, there exists a homomorphically complete class of automata for the α_0^λ-product which contains automata having 2 input signs. On the other hand no class $\mathscr{K}$ consisting of automata having a single input sign can be homomorphically complete for the α_0^λ-product since every automaton in $\mathscr{K}$ would be commutative. Consequently, $S(\mathbf{A})$ would be commutative for each $\mathbf{A}\in\mathscr{K}$, henceforth neither condition (ii) nor (iii) of Corollary 2 could be satisfied by $\mathscr{K}$. Or even, every homomorphically complete class for the α_0^λ-product must contain an infinite number of automata having at least 2 input signs.

Corollary 3. There exists no minimal homomorphically complete class of automata for the α_0^λ-product.

Proof. Suppose that $\mathscr{K}$ is homomorphically complete for the α_0^λ-product. Then $\mathscr{K}$ contains an automaton $\mathbf{B}_0$ which is not counter-free, and there are $\mathbf{B}_1, \ldots, \mathbf{B}_n\in\mathscr{K}$ such that $\mathbf{A}_0\in\mathbf{HSP}^\lambda_{\alpha_0}(\{\mathbf{B}_1, \ldots, \mathbf{B}_n\})$. Since every simple group is isomorphic to a subgroup of a larger simple group, also $\mathscr{K}-\{\mathbf{B}\}$ is homomorphically complete for the α_0^λ-product for any $\mathbf{B}\in\mathscr{K}-\{\mathbf{B}_0, \ldots, \mathbf{B}_n\}$.

Corollary 4. There exists a class of automata which is homomorphically complete for the α_0^λ-product but not homomorphically complete for the α_0-product. Similarly, there is a homomorphically complete class for the α_0^*-product which is not homomorphically complete for the α_0^λ-product.

Proof. By a result of P. Dömösi, there exists a minimal homomorphically complete class of automata for the α_0-product (cf. [1]). Thus, the first statement follows by comparing this result with the previous corollary. To prove the second statement, we give a class $\mathscr{K}$ homomorphically complete for the α_0^*-product but not homomorphically complete for the α_0^λ-product.

For every integer $n\geqq 2$, let $\mathbf{A}_n=([2n]\cup\{2'\}, \{x_1, x_2, x_3, x_4\}, \delta_n)$ be the automaton with transitions $\delta_n(i, x_1)=i+1$ if i is odd, $\delta_n(i, x_2)=i+1 \bmod 2n$ if i is even, $\delta_n(1, x_3)=2$, $\delta_n(2, x_4)=3$, $\delta_n(3, x_3)=2'$, $\delta_n(2', x_4)=1$, and finally, $\delta_n(i, x)=i$,

$\delta_n(2', x)=2'$ in all remaining cases. Put $\mathscr{K}=\{\mathbf{A}_0\}\cup\{\mathbf{A}_n|n\geqq 1\}$. To show that $\mathscr{K}$ is homomorphically complete for the α_0^*-product observe that all automata $\mathbf{B}_n=([n], \{x_1, x_2, x_3\}, \delta_n')$ $(n\geqq 1)$ are in $\mathbf{ISP}^*_{1\alpha_0}(\mathscr{K})$ [2] where δ_n' is defined so that x_1 induces the cyclic permutation $(12\ldots n)$, y the transposition (12), while x_3 induces the identity permutation (1). Thus, $\mathbf{HSP}^*_{\alpha_0}(\mathscr{K})=\mathbf{HSP}^*_{\alpha_0}(\{\mathbf{A}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots\})$ is the class of all automata. On the other hand $\mathscr{K}$ is counter-free, hence $\mathscr{K}$ is not homomorphically complete for the α_0^λ-product.

Before turning to α_1^λ-products we need a few definitions.

A cycle in an automaton (A, X, δ) is a sequence of pairwise distinct states $a_1, \ldots, a_n$ so that $\delta(a_i, x_i)=a_{i+1}$ $(i=1, \ldots, n-1)$ and $\delta(a_n, x_n)=a_1$ for some $x_1, \ldots, x_n\in X$. The integer n is called the length of the cycle. Cycles of length 1 are called trivial, and an automaton is said to be monotone if and only if it contains only trivial cycles. An automaton (A, X, δ) will be called discrete if $\delta(a, x)=a$ for every $a\in A$, $x\in X$. Finally, one-state automata will be referred to as trivial automata.

In the sequel we shall need

Lemma 3. Suppose that an automaton $\mathbf{A}=(A, X, \delta)$ contains a cycle of length at least 2. Then $\mathbf{A}_0\in\mathbf{HSP}^\lambda_{\alpha_1}(\{\mathbf{A}\})$.

Proof. Let us assume that $\mathbf{A}$ contains the nontrivial cycle $a_1, \ldots, a_n$ so that $\delta(a_i, x_i)=a_{i+1}$ $(i=1, \ldots, n-1)$ and $\delta(a_n, x_n)=a_1$ for some $x_1, \ldots, x_n\in X$.

Construct the α_1^λ-product $\mathbf{B}=\mathbf{A}^{n+2}(\varphi, \{x, y\})$, where

$$\varphi_i(c_1, \ldots, c_i, x)=\begin{cases} x_j & \text{if } c_i=a_j\neq a_1,\\ x_1 & \text{if } c_i=a_1 \text{ and } c_m\neq a_1 \text{ when } 1\leqq m<i,\\ \lambda & \text{in all other cases,}\end{cases}$$

$$\varphi_i(c_1, \ldots, c_i, y)=\begin{cases} x_j & \text{if } c_i=a_j\neq a_1,\\ x_1 & \text{if } c_i=a_1 \text{ and } c_m=c_l=a_1 \text{ for some } 1\leqq m<l<i,\\ \lambda & \text{in all other cases.}\end{cases}$$

Taking the subset
$C=\{(c_1, \ldots, c_{n+2})|\{a_2, \ldots, a_n\}\subset\{c_1, \ldots, c_{n+2}\}$ and a_1 is contained exactly 3 times in the system $\{c_1, \ldots, c_{n+2}\}\}$, the automaton $\mathbf{C}=(C, \{x, y\}, \delta_B)$ is a subautomaton of $\mathbf{B}$. Lastly, it can easily be verified that the reset automaton $\mathbf{A}_0$ is a homomorphic image of $\mathbf{C}$ under the mapping $h: C\to[2]$ defined by

$$h(c_1, \ldots, c_{n+2})=$$

$$=\begin{cases}1 & \text{if } a_2 \text{ preceeds at least two occurrences of } a_1 \text{ in } (c_1, \ldots, c_{n+2}),\\ 2 & \text{in all other cases.}\end{cases}$$

Theorem 4. Suppose that $\mathscr{K}$ contains an automaton which is not monotone, and let $\mathbf{A}$ be an arbitrary automaton. Then $\mathbf{A}\in\mathbf{HSP}^\lambda_{\alpha_1}(\mathscr{K})$ $(\mathbf{A}\in\mathbf{HSP}^*_{\alpha_1}(\mathscr{K}))$ if and only if, whenever a simple group G is a homomorphic image of a subgroup of $S(\mathbf{A})$, there exists an automaton $\mathbf{B}\in\mathbf{P}^\lambda_{1\alpha_1}(\mathscr{K})$ $(\mathbf{B}\in\mathbf{P}^*_{1\alpha_1}(\mathscr{K}))$ such that a subgroup of $S(\mathbf{B})$

[2] $P^*_{1\alpha_i}(\mathscr{K})$ denotes the class of all single factor α_i^*-products of automata from $\mathscr{K}$. The operators $P^\lambda_{1\alpha_i}$ and $P_{1\alpha_i}$ are defined similarly.

can be mapped homomorphically onto G. Otherwise, i.e. if $\mathscr{K}$ consists of monotone automata, equation $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})=\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K})$ is universally valid, and 3 cases arise.

(i) If there is a nondiscrete automaton in $\mathscr{K}$, then $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$ is the class of all monotone automata.

(ii) If every automaton from $\mathscr{K}$ is discrete but $\mathscr{K}$ contains a nontrivial automaton, $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$ is the class of all discrete automata.

(iii) Finally, if $\mathscr{K}$ contains only trivial automata, then $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$ is the class of all trivial automata.

Proof. Assume that $\mathscr{K}$ contains a nonmonotone automaton. Then $\mathbf{P}^{\lambda}_{1\alpha_1}(\mathscr{K})$ is not counter-free and $\mathbf{A}_0\in\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$. Since $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})=\mathbf{HSP}_{\alpha_0}\mathbf{P}^{\lambda}_{1\alpha_1}(\mathscr{K})=$ $=\mathbf{HSP}^{\lambda}_{\alpha_0}\mathbf{P}^{\lambda}_{1\alpha_1}(\mathscr{K})$, the first statement of Theorem 4 follows by Theorem 3 for α^{λ}_0-products. As regards α^{*}_1-products, the proof is similar just use equation $\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K})=$ $=\mathbf{HSP}^{\lambda}_{\alpha_0}\mathbf{P}^{*}_{1\alpha_1}(\mathscr{K})$.

Now suppose that $\mathscr{K}$ contains only monotone automata. Then the same holds for $\mathscr{K}^{*}$, and by $\mathbf{HSP}^{*}_{g}(\mathscr{K})=\mathbf{HSP}_{g}(\mathscr{K}^{*})$, even for $\mathbf{HSP}^{*}_{g}(\mathscr{K})$.

If there is a nondiscrete automaton in $\mathscr{K}$, then the elevator $\mathbf{E}=([2], \{x, y\}, \delta)$ having transitions $\delta(1, x)=1$, $\delta(1, y)=\delta(2, x)=\delta(2, y)=2$ is in $\mathbf{IP}^{\lambda}_{1\alpha_0}(\mathscr{K})$. By a result in [7], every monotone automaton is already in $\mathbf{ISP}_{\alpha_0}(\{\mathbf{E}\})$. Hence we have $\mathbf{HSP}^{*}_{g}(\mathscr{K})=\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K})=\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})=\mathbf{ISP}_{\alpha_0}\mathbf{P}^{\lambda}_{1\alpha_0}(\mathscr{K})=\mathbf{ISP}^{\lambda}_{\alpha_0}(\mathscr{K})$ is the class of all monotone automata.

The proof in the remaining two cases is obvious. We have $\mathbf{HSP}^{*}_{g}(\mathscr{K})=\mathbf{ISP}_{\alpha_0}(\mathscr{K})$.

Corollary 5. There exists an algorithm to decide for a finite class $\mathscr{K}$ and an automaton $\mathbf{A}$ whether $\mathbf{A}\in\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$ $(\mathbf{A}\in\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K}))$.

Corollary 6. Since $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})\subseteqq\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K})$ always holds, $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})=$ $=\mathbf{HSP}^{*}_{\alpha_1}(\mathscr{K})$ if and only if one of the following 2 conditions is valid.

(i) $\mathscr{K}$ consists of monotone automata.

(ii) There is a nonmonotone automaton in $\mathscr{K}$, and whenever a simple group G is a homomorphic image of a subgroup of $S(\mathbf{A})$ for an automaton $\mathbf{A}\in\mathbf{P}^{*}_{1\alpha_1}(\mathscr{K})$, there is an automaton $\mathbf{B}\in\mathbf{P}^{\lambda}_{1\alpha_1}(\mathscr{K})$ such that a subgroup of $S(\mathbf{B})$ can be mapped homomorphically onto G.

Corollary 7. A class $\mathscr{K}$ of automata is homomorphically complete for the α^{λ}_1-product (α^{*}_1-product) if and only if, for every simple group G, there exists an automaton $\mathbf{A}\in\mathbf{P}^{\lambda}_{1\alpha_1}(\mathscr{K})$ $(\mathbf{A}\in\mathbf{P}^{*}_{1\alpha_1}(\mathscr{K}))$ so that a subgroup of $S(\mathbf{A})$ can be mapped homomorphically onto G.

Corollary 8. There exists no minimal homomorphically complete class for the α^{λ}_1-product (α^{*}_1-product).

Now we present a new proof for a part of a nice result of F. Gécseg [7].

Theorem 5. The following 3 statements are equivalent for every class $\mathscr{K}$ of automata.

(i) $\mathscr{K}$ is homomorphically complete for the α^{*}_1-product.

(ii) For every integer $n\geqq 1$, there exists an automaton $\mathbf{A}_n=(A, X, \delta)\in\mathscr{K}$

having at least n different states $a_1, \ldots, a_n \in A$ such that for every $i, j \in [n]$, there is a word $p \in X^*$ satisfying $\delta(a_i, p) = a_j$.

(iii) $\mathscr{K}$ is isomorphically complete for the α_1^*-product.

Proof. We prove that (i) implies (ii). Suppose that $\mathscr{K}$ is homomorphically complete for the α_1^*-product. It is enough to prove (ii) for n prime. Take the cyclic group Z_n. Since Z_n is simple, there are an automaton $\mathbf{A}_n' = (A, X', \delta') \in \mathbf{P}^*_{1\alpha_1}(\mathscr{K})$ and a subgroup H of $S(\mathbf{A}_n')$ such that Z_n is a homomorphic image of H. Note that H is isomorphic to a permutation group of a subset $A' \subseteq A$. Since Z_n has an element of order n, there must be a translation $t_p \in H$ of order kn for an integer $k \geqq 1$. Henceforth, there are different states $a_1, \ldots, a_{ln} \in A'$ $(l \geqq 1)$ for which $\delta(a_i, p) = a_{i+1 \bmod ln}$ $(i \in [kn])$. Taking $a_1, \ldots, a_n$ we see that $\mathbf{A}_n'$ satisfies condition (ii). Let $\mathbf{A}_n = (A, X, \delta) \in \mathscr{K}$ be an automaton such that $\mathbf{A}_n' \in \mathbf{P}^*_{1\alpha_1}(\mathbf{A}_n)$. Clearly, also $\mathbf{A}_n$ satisfies (ii) with $a_1, \ldots, a_n \in A$.

For the sake of completeness we recall from [7] that every n-state automaton is already in $\mathbf{ISP}^*_{1\alpha_1}(\{\mathbf{A}_n\})$, while (iii)$\Rightarrow$(i) is trivial.

Suppose we are given $n \geqq 1$ boxes $B_1, \ldots, B_n$ and $k \leqq n$ pebbles numbered from 1 to k. In addition, k boxes, say $B_{i_1}, \ldots, B_{i_k}$, are distinguished so that $i_1 < \ldots < i_k$. Initially B_{i_j} contains the pebble numbered j, $j = 1, \ldots, k$. The game goes on as follows. At each step we take out the pebbles from the boxes and put all pebbles which were in B_i back into B_i or put all of them into box B_{i+1}. The pebbles from B_n go into B_n or B_1. After a number of steps the pebbles get back into the distinguished boxes, each distinguished box B_{i_t} containing a pebble numbered j_t, $t \in [k]$. Clearly, $(j_1 \ldots j_k)$ is a power of the cyclic permutation $(1 \ldots k)$. This proves our

Observation. Let $\mathbf{C}_n$ be a counter, $\mathbf{A} \in \mathbf{P}^{\lambda}_{1\alpha_1}(\mathbf{C}_n)$. Then every subgroup of $S(\mathbf{A})$ is isomorphic to a subgroup of a cyclic group Z_k with $k \leqq n$, whence cyclic.

Corollary 9. There exists a class $\mathscr{K}$ which is homomorphically complete for the α_i^*-product but not homomorphically complete for the α_1^{λ}-product.

Proof. Take a class $\mathscr{K}$ consisting of a counter $\mathbf{C}_n$ for each $n \geqq 1$. $\mathscr{K}$ is homomorphically complete for the α_1^*-product by Theorem 5. Since every subgroup of $S(\mathbf{C}_n)$ $(n \geqq 1)$ is cyclic, but there are noncyclic finite simple groups, $\mathscr{K}$ is not homomorphically complete for the α_1^{λ}-product.

We do not know whether there exists a class $\mathscr{K}$ which is homomorphically complete for the α_1^{λ}-product but not homomorphically complete for the α_1-product[3]. It is clear that there exists a class $\mathscr{K}$ such that $\mathbf{HSP}_{\alpha_1}(\mathscr{K})$ is a proper subclass of $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$, take e.g. $\mathscr{K} = \{([2], \{x\}, \delta)\}$, $\delta(1, x) = \delta(2, x) = 2$.

Now we turn our attention to the α_2^{λ}-product and the g^{λ}-product.

Theorem 6. $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K}) = \mathbf{HSP}^*_g(\mathscr{K})$ for every class $\mathscr{K}$. Furthermore, four cases arise. If $\mathscr{K}$ contains a nonmonotone automaton, then $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})$ is the class of all automata. If $\mathscr{K}$ consists of monotone automata one of which is not discrete, then $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})$ is the class of all monotone automata. If $\mathscr{K}$ consists of discrete automata and contains a nontrivial automaton, then $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})$ is the class of all

[3] Recently Ésik has shown the existence of such a class.

discrete automata. Finally, if every automaton contained by $\mathscr{K}$ is trivial, then $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})$ is the class of all trivial automata.

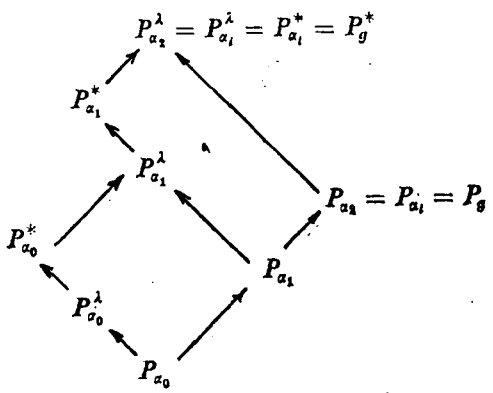
Proof. First we recall a result proved in [4]. If $\mathscr{K}$ is a class of automata such that there is an automaton $(A, X, \delta)\in\mathscr{K}$ having a state $a\in A$, signs $x, y\in X$ and words $p, q\in X^*$ with $\delta(a, x)\neq\delta(a, y)$ and $\delta(a, xp)=\delta(a, yq)=a$, then $\mathbf{HSP}_{\alpha_2}(\mathscr{K})$ is the class of all automata.

Now suppose that $\mathscr{K}$ contains an automaton which is not monotone. Then $\mathscr{K}^{\lambda}$ is homomorphically complete for the α_2-product. Hence, $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})=\mathbf{HSP}_{\alpha_2}(\mathscr{K}^{\lambda})$ is the class of all automata, and since $\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})\subseteq\mathbf{HSP}^{*}_{g}(\mathscr{K})$, the same is true for $\mathbf{HSP}^{*}_{g}(\mathscr{K})$.

For the proof of the remaining cases see Theorem 4.

Corollary 10. There exists an algorithm to decide for a finite class $\mathscr{K}$ and an automaton $\mathbf{A}$ if $\mathbf{A}\in\mathbf{HSP}^{\lambda}_{\alpha_2}(\mathscr{K})$.

Now we come to the point of comparing the strengths of our various products with respect to homomorphic realization. The following figure gives a summary. The figure is to be interpreted as follows. If two operators, say $\mathbf{P}$ and $\mathbf{Q}$ label the same node, then this expresses that $\mathbf{P}$ and $\mathbf{Q}$ are homomorphically equivalent, i.e. $\mathbf{HSP}(\mathscr{K})=\mathbf{HSQ}(\mathscr{K})$ for every class $\mathscr{K}$. If there is a directed path from a node labeled $\mathbf{P}$ to a node labeled $\mathbf{Q}$ then $\mathbf{Q}$ is homomorphically more general than $\mathbf{P}$. This means that $\mathbf{HSP}(\mathscr{K})\subseteq\mathbf{HSQ}(\mathscr{K})$ for every $\mathscr{K}$, but there exists a class for which the inclusion is proper. Further on this situation will be denoted by $\mathbf{P}<\mathbf{Q}$. The index i denotes an arbitrary integer exceeding 2.



To justify the correctness of this figure first observe that all equivalences have been proved previously except for that the α_2-product is homomorphically equivalent to the general product. But this is the main result of [5]. On the other hand, all relations $\mathbf{P}<\mathbf{Q}$ appearing in the diagram have been established in this paper or in several papers earlier (cf. e.g. [7], [8], [9]), the only exception being $\mathbf{P}^{*}_{\alpha_0}<\mathbf{P}^{\lambda}_{\alpha_1}$.

To prove $\mathbf{HSP}^{*}_{\alpha_0}(\mathscr{K})\subseteq\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathscr{K})$ for arbitrary $\mathscr{K}$, let us distinguish 3 cases.

Case 1. $\mathcal{K}$ contains a nonmonotone automaton. Then the inclusion follows by Corollary 1 and Theorem 4.

Case 2. $\mathcal{K}$ consists of monotone automata, one of which is not discrete. In this case we have $\mathbf{HSP}^*_{\alpha_0}(\mathcal{K})=\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathcal{K})$ equals the class of all monotone automata. (Hint: an automaton in $\mathbf{IS}(\mathcal{K}^{\lambda})$ is X-isomorphic to $\mathbf{E}$.)

Case 3. $\mathcal{K}$ consists of discrete automata. Now we have $\mathbf{ISP}_{\alpha_0}(\mathcal{K})=\mathbf{HSP}^*_g(\mathcal{K})$, thus, $\mathbf{HSP}^*_{\alpha_0}(\mathcal{K})=\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathcal{K})$.

On the other hand, $\mathbf{HSP}^*_{\alpha_0}(\mathcal{K})$ is properly contained by $\mathbf{HSP}^{\lambda}_{\alpha_1}(\mathcal{K})$ e.g. for $\mathcal{K}=\{\mathbf{C}_2^{\lambda}\}$.

It should be noted that no more arrows could be added to the diagram.

3. A homomorphically complete class for the α_0-product

It was pointed out in Example 1 that there exists a class of automata having 2 input signs homomorphically complete for the α_0^{λ}-product. Our principal goal in this section is to show that this result can be strengthened. Such a class does exist for the α_0-product as well. This is interesting because we do not know any direct way for proving that the class of all automata with 2 input signs is homomorphically complete for the α_0-product.

Let $\mathbf{A}=(A, X, \delta)$ be an arbitrary automaton, and take a subsemigroup S of $S(\mathbf{A})$ containing an identity element. Put $\mathbf{A}^S=(A^S, S, \delta^S)$, where $A^S=\{b\in A \mid \exists a\in A, t\in S\ b=t(a)\}$ and $\delta^S(a, t)=t(a)$ for any $a\in A^S$, $t\in S$. Observe that letting $S=S(\mathbf{A})$ we get back the definition of $\mathbf{A}^*$.

The following generalization of Theorem 2 is straightforward.

Theorem 7. Let $\mathbf{A}=(A, X, \delta)$ be an automaton, S a subsemigroup of $S(\mathbf{A})$ containing identity element. Assume that there exists an integer $n\geqq 1$ satisfying $S\subseteq\{t_p^{\mathbf{A}} \mid p\in X^n\}$. Then $\mathbf{A}^S\in\mathbf{HSP}_{\alpha_0}(\{\mathbf{C}_n, \mathbf{A}_0, \mathbf{A}\})$.

The characteristic semigroup of $\mathbf{A}^S$ is isomorphic to S. Let $\mathbf{B}=(B, Y, \delta)$ be an arbitrary automaton. We may construct the automaton $\mathbf{B}'=(S(B), Y, \delta')$ with transitions $\delta'(t_p^{\mathbf{B}}, y)=t_{py}^{\mathbf{B}}$, $p\in Y^*$, $y\in Y$. It is well-known that $\mathbf{B}'$ is isomorphic to a subautomaton of a direct power of $\mathbf{B}$. Henceforth $\mathbf{B}'\in\mathbf{HSP}_{\alpha_0}(\{\mathbf{B}\})$, and we have

Corollary 11. Under the assumptions of Theorem 7 it follows that $\mathbf{A}_S=(S, S, \delta_S)\in\mathbf{HSP}_{\alpha_0}(\{\mathbf{C}_n, \mathbf{A}_0, \mathbf{A}\})$ where $\delta_S(s_1, s_2)=s_1s_2$, $s_1, s_2\in S$.

Suppose now a class $\mathcal{K}$ of automata satisfies the following list of conditions.

(i) $\mathbf{A}_0\in\mathbf{HSP}_{\alpha_0}(\mathcal{K})$.

(ii) There exist an automaton $\mathbf{B}_0\in\mathcal{K}$, a subsemigroup $S_0\subseteq S(\mathbf{B}_0)$ isomorphic to $S(\mathbf{A}_0^{\lambda})$ so that for some n, an n-state counter $\mathbf{C}_n$ is in $\mathbf{HSP}_{\alpha_0}(\mathcal{K})$ and, at the same time, all elements of S_0 are induced by words of length n.

(iii) For every finite simple group G there exist an automaton $\mathbf{B}_G\in\mathcal{K}$, a subgroup $H_G\subseteq S(\mathbf{B}_G)$, and an integer $n\geqq 1$ satisfying

(iii_1) H_G can be mapped homomorphically onto G,

(iii_2) $\mathbf{C}_n\in\mathbf{HSP}_{\alpha_0}(\mathcal{K})$,

(iii_3) every element of H_G is induced by a word of length n.

Set $\mathscr{K}' = \{\mathbf{A}_{S_0}, \mathbf{A}_{H_G} | G$ is a finite simple group$\}$. Since $\mathscr{K}'$ is obviously not counter-free, $\mathbf{A}_{S_0}$ is X-isomorphic to $\mathbf{A}_0^\lambda$, finally, the characteristic semigroup of $\mathbf{A}_{H_G}$ is isomorphic to H_G, Corollary 11 yields that $\mathbf{HSP}_{\alpha_0}^\lambda(\mathscr{K}')$ is the class of all automata. Since every automaton belonging to $\mathscr{K}'$ has an input sign inducing the identity state-map, $\mathbf{HSP}_{\alpha_0}^\lambda(\mathscr{K}') = \mathbf{HSP}_{\alpha_0}(\mathscr{K}')$. However $\mathbf{HSP}_{\alpha_0}$ is a closure operator, thus $\mathscr{K}$ is homomorphically complete for the α_0-product. This is the basis of our last result.

Theorem 8. There exists a class of automata having 2 input signs which is homomorphically complete for the α_0-product.

Proof. Let $\mathbf{B}_0 = ([2], \{x, y\}, \delta_0)$ be the automaton with transitions $\delta_0(1, x) = 2$, $\delta_0(1, y) = \delta_0(2, x) = \delta_0(2, y) = 1$. Translations $t_{xx}^{\mathbf{B}_0}$, $t_{xy}^{\mathbf{B}_0}$, $t_{yx}^{\mathbf{B}_0}$ form a subsemigroup S_0 of $S(\mathbf{B}_0)$ isomorphic to $S(\mathbf{A}_0^\lambda)$ under the correspondence $t_{xx}^{\mathbf{B}_0} \to t_{x_0}^{\mathbf{A}_0^\lambda}$, $t_{xy}^{\mathbf{B}_0} \to t_x^{\mathbf{A}_0^\lambda}$, $t_{yx}^{\mathbf{B}_0} \to t_y^{\mathbf{A}_0^\lambda}$. In addition, for every odd integer $n \geqq 3$, take the automaton $\mathbf{B}_n = ([n], \{x, y\}, \delta_n)$ so that x induces the transposition (12) and y induces the cyclic permutation $(1 \ldots n)$. Besides, since n is odd, there exists an odd integer m satisfying $S_n = \{t_p^{\mathbf{B}_n} | p \in \{x, y\}^m\}$. As a matter of fact, there is an m' such that every permutation of $[n]$ can be induced by a word of length at most m'. Put m the least odd integer not less than $m'+n$. Let $t = t_{p_n}^{\mathbf{B}}$, $|p| = k \leqq m'$. If $m-k$ is even, put $q = py^{m-k}$. If $m-k$ is odd, take $q = px^n y^{m-(k+n)}$. We have $t = t^{\mathbf{B}_n}$ in both cases.

Now set

$$\mathscr{K} = \{\mathbf{A}_0, \mathbf{B}_0, \mathbf{B}_n | n \geqq 3 \text{ is odd}\}.$$

Since all counters of odd length as well as $\mathbf{C}_2$ are obviously in $\mathbf{HSP}_{\alpha_0}(\mathscr{K})$ and every finite simple group is isomorphic to a subgroup of S_n for odd n, $\mathscr{K}$ is homomorphically complete for the α_0-product. It should be noted that from the proof of Lemma 3 we have that $\mathbf{A}_0$ can be omitted from $\mathscr{K}$.

Corollary 12. The class $\mathscr{K}$ consisting of $\mathbf{A}_0^\lambda$ and automata $\mathbf{A}_n = ([n], \{x, y\}, \delta_n)$ $(n \geqq 3)$ with $t_x = (1 \ldots n)$, $t_y = (12)$ is homomorphically complete for the α_0-product. Recall that the main result of Dömösi's paper is the homomorphic completeness of $\mathscr{K}^\lambda$ for the α_0-product.

A. JÓZSEF UNIVERSITY
BOLYAI INSTITUTE
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] Dömösi, P., On minimal R-complete systems of finite automata, Acta Cybernetica 3 (1976), 37—41.
[2] Dömösi, P., Candidate Dissertation, Budapest, 1981.
[3] Dömösi, P., On cascade products of standard automata, Automata, Languages and Mathematical Systems, Proc. Conf. Salgótarján, 1984, 37—45.
[4] Ésik, Z., Homomorphically complete classes of automata with respect to the α_2-product, Acta Sci. Math., 48 (1985) 135—141.
[5] Ésik, Z. and Gy. Horváth, The α_2-product is homomorphically general, Papers on Automata Theory, K. Marx Univ. of Economics, Dept. of Math., V (1983), 49—62.

[6] Евтушенко, Н. В., К реализации автоматов каскадным соединением стандартных автоматов, Автоматика и вычислительная техника 1979, №. 2. 50—53.
[7] GÉCSEG, F., On products of abstract automata, Acta Sci. Math., 38 (1976), 21—43.
[8] GÉCSEG, F., Composition of automata, Automata, Languages and Programming, 2nd colloq., Saarbrücken, 1974, LNCS 14, 351—368.
[9] GÉCSEG, F., Products of automata, manuscript to be published by Springer-Verlag, 1986.
[10] Глушков, В. М., Абстрактная теория автоматов, Успехи матем. наук, 16:5(101), 1961, 3—62.
[11] IMREH, B., On finite definite automata, Acta Cybernetica, 7 (1985), 61—65.

(Received March 6, 1985)

Properties of the fuzzy connectives in the light of the general representations theorem

By J. DOMBI

Introduction

A number of papers deal with the operators of fuzzy logic. All of these publications assume associativity, isotonicity, continuity, and the validity of the permanence principle. The representation theorem of the class of such operators were examined only under assuming strict isotonicity in the literature [2], [6], [7], [8]. Here we shall consider properties of the operator class supposing isotonicity only, with conditions under which De Morgan identity holds; first for the case with Archimedean properties, and then for the general one. The properties of the limes operators of operator series will be studied, too. Finally, the distributive class is determined.

Preliminaries and basic results

Concerning the conjunction $c(x, y)$ and disjunction $d(x, y)$ featuring if fuzzy logic, we assume that the mappings $c: [0, 1]\times[0, 1]\to[0, 1]$ and $d: [0, 1]\times[0, 1]\to [0, 1]$ are:

(i) associative:

$$c(x, c(y, z)) = c(c(x, y), z), \tag{1}$$

$$d(x, d(y, z)) = d(d(x, y), z).$$

(ii) isotonic, i.e. if $x\leqq x'$, then for all y:

$$c(x, y) \leqq c(x', y), \quad d(x, y) \leqq d(x,' y), \tag{2}$$

$$c(y, x) \leqq c(y, x'), \quad d(y, x) \leqq d(y, x'). \tag{3}$$

(iii) the principle of permanence holds:

$$c(0, 0) = c(0, 1) = c(1, 0) = 0, \quad c(1, 1) = 1, \tag{4}$$

$$d(1, 1) = d(0, 1) = d(1, 0) = 1, \quad d(0, 0) = 0. \tag{5}$$

(iv) continuous.

Since the theorems related to the conjunctive operator can be proved in an analogous way as those related to the disjunction, the theorems below are proved only for conjunctive operators.

Theorem 1. For a conjunctive operator (disjunctive operator) satisfying conditions (i—iv) it holds that

$$c(x, 1) = c(1, x) = x, \quad d(x, 1) = d(1, x) = 1, \tag{6}$$

$$c(x, 0) = c(0, x) = 0, \quad d(x, 0) = d(0, x) = x. \tag{7}$$

Proof. [3].

Definition. The conjunctive operator $c(x, y)$ (disjunctive operator $d(x, y)$) is Archimedean if $c(x, y) < x$ and $d(x, y) > x$ for $x \in (0, 1)$.

Definition. The pseudo-inverse of a strictly monotonously decreasing function $f : [a, b] \to [f(a), f(b)]$ is

$$f^{(-1)}(x) = \begin{cases} b & \text{if } x \leqq f(b), \\ f^{-1}(x) & \text{if } f(b) \leqq x \leqq f(a), \\ a & \text{if } f(a) \leqq x. \end{cases} \tag{8}$$

Theorem 2. For an Archimedean conjunctive operator (disjunctive operator) with properties (i—iv) there always exists a strictly monotonous function f_c (f_d), for which

$$f_c(1) = 0, \quad f_d(0) = 0, \tag{9}$$

$$f_c(0) = r_c, \quad f_d(1) = r_d, \tag{10}$$

where r_c and r_d may be $+\infty$ or $-\infty$, and is such that

$$c(x, y) = f_c^{(-1)}(f_c(x) + f_c(y)), \quad d(x, y) = f_d^{(-1)}(f_d(x) + f_d(y))$$

where, apart from a factor $\alpha \neq 0$, $f_c(x)$ $(f_d(x))$ is uniquely defined.

The function $f_c(x)$ $(f_d(x))$ will be called "the additive generator of the function $c(x, y)$ $(d(x, y))$".

Proof. [9].

Consequence. All such operators are commutative.

Definition. The operator $c(x, y)$ $(d(x, y))$ is reducible with respect to both sides if, in the case $(t, w] \in (0, 1]$ $((t, w] \in [0, 1))$ $c(t, u) = c(t, v)$ or $c(u, w) = c(v, w)$ $(d(t, u) = d(t, v)$ or $d(u, w) = d(v, w))$ holds if and only if, $u = v$. If $c(x, y)$ $(d(x, y))$ is continuous, then the reducibility is equivalent to the strict isotonicity.

In the particular case when the operation $c(x, y)$ $(d(x, y))$ is strictly isotonic, we may obtain from Theorem 2 the theorem of Aczél [1], and then $r_c = -\infty$ or $r_c = \infty$.

A consideration of this special case is to be found in [3]. Let us subsequently assume that $r_c \neq \pm\infty$, $r_d \neq \pm\infty$.

Theorem 3. For the monotonously decreasing additive generator $f_c(x)$ $(f_d(x))$ of $c(x, y)$ $(d(x, y))$, there always exists a strictly monotonously increasing additive generator of $c(x, y)$.

Proof. Since the additive generator $f_c(x)$ is determined up to a constant multiple factor $\alpha \neq 0$, let us select $\alpha = -1$ and define

$$\tilde{f}_c(x) = -f_c(x);$$

this $\tilde{f}_c$ will then satisfy the statement of the theorem.

We restrict our considerations below to monotonously decreasing generator when we are speaking on conjunctions, and to monotonously increasing additive one in the case of disjunctions, that is, we shall suppose that

$$0 < r_c < \infty \quad (0 < r_d < \infty)$$

Theorem 4. For the additive generator $f_c(x)\,(f_d(x))$ of the operation $c(x, y)$ $(d(x, y))$ we can give a multiplicative generator $\tilde{f}_c(x)\,(\tilde{f}_d(x))$, where $f_{c_r}(x): [0, 1] \to$

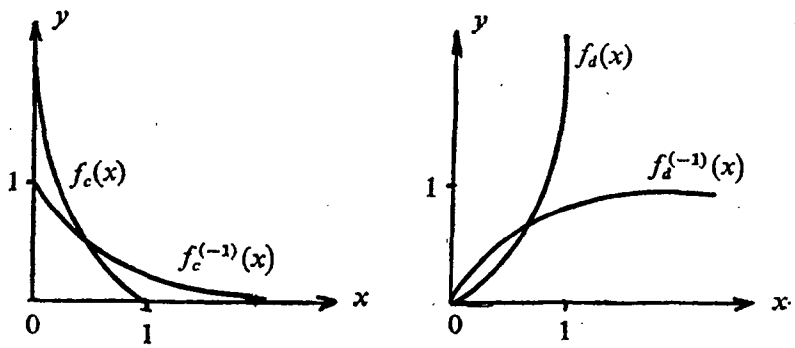


Fig. 1
Generator functions of coujunctive and disjunctive operators for the additive representation case

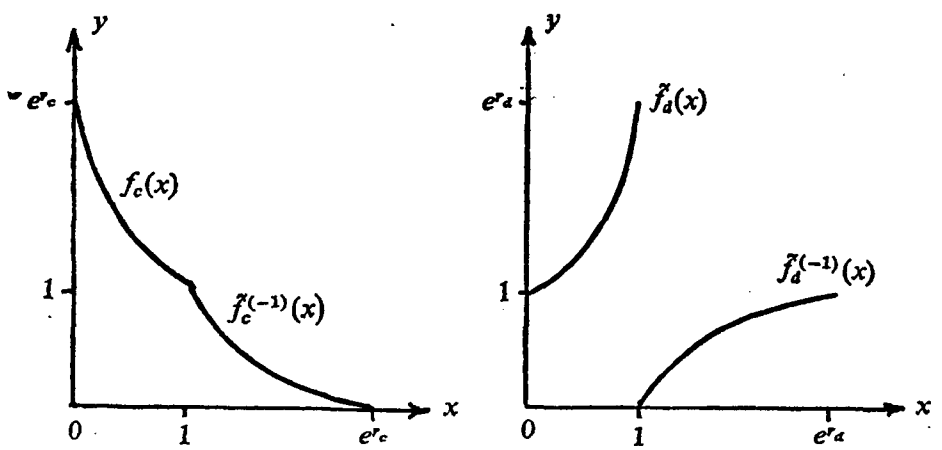


Fig. 2
Generator functions of conjunctive and disjunctive operators for the multiplicative representation case

$\to[e^{r_c}, 1]$, $f_d(x)\colon [0, 1] \to [1, e^{r_d}]$,

$$c(x, y) = \tilde{f}_c^{(-1)}(\tilde{f}_c(x)\tilde{f}_c(y)), \quad d(x, y) = f_d^{(-1}(\tilde{f}_d(x)f_d(y)). \tag{11}$$

Proof. The proof is obvious if we put

$$\tilde{f}_c(x) = \exp(f_c(x)), \quad \text{so: } f_c^{(-1)}(x) = f_c^{(-1)}(\ln(x))$$

and

$$\tilde{f}_c^{(-1}(\tilde{f}_c(x)\,\tilde{f}_c(y)) = f_c^{(-1)}(\ln(\exp(f_c(x))) \cdot \exp(f_c(y))) = f_c^{(-1)}(f_c(x)+f_c(y)).$$

3. The De Morgan class of fuzzy operators

For investigation of De Morgan identity we need a mapping, the so called negation $n\colon [0, 1]\to[0, 1]$. We shall assume the usual properties:

(1) n decreases strictly monotonously,
(2) n is continuous,
(3) the principle of permanence holds for n

$$n(1) = 0 \quad \text{and} \quad n(0) = 1.$$

Below, we distinguish two forms of De Morgan identity, the conjunctive form

$$n(c(x, y)) = d(n(x), n(y)) \tag{12}$$

and the disjunctive form

$$n(d(x, y)) = c(n(x), n(y)). \tag{13}$$

The theorems will be stated and proved only for the conjunctive form.

Theorem 5. For a given conjunctive (disjunctive) operator $c(x, y)$ $(d(x, y))$ and a negation operator $n(x)$, it is possible to construct a disjunctive (conjunctive) operator satisfying the conjunctive (disjunctive) De Morgan identity.

Proof. Let $f_c(x)$ be the generator function of $c(x, y)$. Let us define

$$f_d(x) = \frac{f_c(n^{-1}(x))}{a} \tag{14}$$

where $a>0$ is an optional real number. The function $d(x, y)$ generated by $f_d(x)$ will in fact be a disjunction, since $f_d(x)$ satisfies the conditions which ensure that $f_d(x)$ is a generator function of a disjunctive operator. Furthermore,

$$n(c(x, y)) = n(f_c^{(-1)}(f_c(x)+f_c(y))) = n\left(f_c^{(-1)}\, a\left(\frac{f_c(n^{-1}(n(x)))}{a}+\frac{f_c(n^{-1}(n(y)))}{a}\right)\right) =$$

$$= n(f_c^{(-1)}\, a(f_d(n(x))+f_d(n(x)))) = f_d^{(-1)}(f_d(n(x))+f_d(n(y))) = d(n(x), n(y))$$

where we have made use of the fact that $n(f_c^{(-1)}(ax))=f_d^{(-1)}(x)$, and thus the theorem is proved.

Theorem 6. Let $c(x, y)$ be a conjunction, and $d(x, y)$ a disjunction. A negation $n(x)$ can then always be given such that the conjunctive (disjunctive) De Morgan identity holds.

Proof. Let

$$n(x) = f_d^{(-1)}\left(\frac{f_d(1)}{f_c(0)} f_c(x)\right) = f_d^{(-1)}\left(\frac{r_d}{r_c} f_c(x)\right) = f_d^{(-1)}(a f_c(x)).$$

As concerns the $n(x)$ defined in this way it can be seen that

a) it is bimorphous, since $\frac{r_d}{r_c} f_c(x) \in [0, r_d]$,

b) it decreases strictly monotonously, because of the monotonicity of $f_c(x)$ and $f_d^{(-1)}(x)$,

c) $n(x)$ is continuous, since $f_c(x)$ and $f_d^{-1}(x)$ are such too,

d) it satisfies the conjunctive De Morgan identity.

From the substitution $x = n^{-1}(y)$ we obtain

$$f_d(y) = \frac{f_c(n^{-1}(y))}{a}$$

from which the statement follows.

Theorem 7. Let $c(x, y)$ be a conjunctive, $d(x, y)$ a disjunctive operator and $n(x)$ a negation. The conjunctive De Morgan identity holds if and only if

$$f_d(x) = \frac{f_c(n^{-1}(x))}{a}.$$

Proof. The sufficiency has already been seen. From the conjunctive De Morgan identity we have

$$d(x, y) = n^{-1}(c(n^{-1}(x), n^{-1}(y))). \tag{15}$$

The conjunctive operator is associative

$$c(c(x, y), z) = c(x, c(y, z)),$$

and so

$$n(c(c(x, y), z)) = n(c(x, c(y, z))).$$

Using the conjunctive De Morgan identity

$$d(n(c(x, y)), n(z)) = d(n(x), n(c(y, z))).$$

Using again the conjunctive De Morgan identity

$$d(d(n(x), n(y)), n(z)) = d(n(x), d(n(y), n(z))).$$

Replacing $n(x)$, $n(y)$, $n(z)$ by x, y, z, we find that $d(x, y)$ is associative.

$d(x, y)$ is isotonic. If $y \leqq y'$, then $n(y) \geqq n(y')$, $c(x, y)$ is isotonic, hence

$$c(n(x), n(y)) \geqq c(n(x), n(y')).$$

As $n^{-1}(x)$ is strictly monotonous

$$d(x, y) = n^{-1}(c(n(x), n(y))) \leqq n^{-1}(c(n(x), n(y'))) = d(x, y').$$

The continuity of $d(x, y)$ is a consequence of those of $c(x, y)$ and $n(x)$.
$d(x, y)$ is Archimedean since

$$c(n(x), n(x)) < n(x) \quad x \in (0, 1),$$

and so

$$d(x, x) = n^{-1}(c(n(x), n(x))), \quad n^{-1}(n(x)) = x.$$

$d(x, y)$ satisfies the permanence principle:

$$c(x, 1) = c(1, x) = x,$$

$$c(x, 0) = c(0, x) = 0,$$

and so

$$d(x, 0) = n^{-1}(c(n(x), n(0))) = n^{-1}(n(x)) = x,$$

$$d(x, 1) = n^{-1}(c(n(x), n(1))) = n^{-1}(0) = 1.$$

Thus, for $d(x, y)$ there exists a generator function $f_d(x)$ which is determined uniquely, apart from a constant factor. We have seen that (14) is a generator function, and since its constant factor too is of such a form, we have proved the theorem.

If $n(n(x)) = x$ holds, then the conjunctive and disjunctive De Morgan identities are the same.

Let us now consider the non-Archimedean case. On the basis of the representative theorem of [9], there exists a finite or (uncountably) infinite series of discrete intervals $M_i = (a_i, b_i)$ such that

$$c(x, y) = \begin{cases} f_{c,i}^{(-1)}(f_{c,i}(x) + f_{c,i}(y)) & x, y \in (a_i, b_i), \\ \min(x, y) & x, y \in [0, 1] \setminus \bigcup_i M_i. \end{cases} \tag{16}$$

The De Morgan class is constructed so that the conditions required in the Archimedean case hold for the generator functions in the corresponding (a_i, b_i).

4. Limes operators of the operator class

An important role is played below by the following definition [6].
Definition: Let

$$t_c(x, y) = \begin{cases} x & \text{if } y = 1, \\ y & \text{if } x = 1, \\ 0 & \text{otherwise,} \end{cases} \qquad t_d(x, y) = \begin{cases} x & \text{if } y = 0, \\ y & \text{if } x = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

be a non-continuous conjunctive or disjunctive operator.

Theorem 8. All conjunctive (disjunctive) operators satisfying conditions (i—iv) have the following properties:

$$t_c(x, y) \leqq c(x, y) \leqq \min(x, y); \quad \max(x, y) \leqq d(x, y) \leqq t_d(x, y). \tag{18}$$

Proof. By definition we have $t_c(x, y) = 0$, $c(x, y) \geqq 0$ provided that $x, y \in [0, 1)$ and $t_c(1, x) = t_c(x, 1) = c(1, x) = c(x, 1) = x$, thus

$$t_c(x, y) \leqq c(x, y).$$

Moreover, let us assume that $x \leqq y$. Utilizing the fact that $f_c(x)$ and $f_c^{(-1)}(x)$ decrease strictly monotonously, we have

$$c(x, y) = f_c^{(-1)}(f_c(x)+f_c(y)) \leqq f_c^{(-1)}(f_c(x)) = x = \min(x, y).$$

Theorem 9. Let $\lambda>0$ be a real. If $f_c(x)\,(f_d(x))$ is the generator function of a conjunctive (disjunctive) operator $c(x, y)\,(d(x, y))$, then

$$f_c^{\lambda}(x)\ (f_d^{\lambda}(x)) \tag{19}$$

are also the generator functions of some conjunction and disjunction denoted by $c_\lambda(x, y)\,(d_\lambda(x, y))$.

Proof. If $f_c(x)$ is the generator function of the conjunctive operator, then $f_c(x)$ too satisfies the properties of the generator function. Thus

$$c_\lambda(x, y) = f_c^{(-1)}((f_c^{\lambda}(x)+f_c^{\lambda}(y))^{1/\lambda});$$
$$d_\lambda(x, y) = f_d^{(-1)}((f_d^{\lambda}(x)+f_d^{\lambda}(y))^{1/\lambda}).$$

Theorem 10. Let λ, $c_\lambda(x, y)$, $d_\lambda(x, y)$ be the same as above. Then the following relations are true.

1. $$\lim_{\lambda\to\infty} c_\lambda(x, y) = \min(x, y), \quad \lim_{\lambda\to\infty} d_\lambda(x, y) = \max(x, y), \tag{20}$$

2. $$\lim_{\lambda\to 0} c_\lambda(x, y) = t_c(x, y), \quad \lim_{\lambda\to 0} d_\lambda(x, y) = t_d(x, y), \tag{21}$$

Proof. 1. Let us assume that $x \leqq y$; then $\min(x, y)=x$. Since it holds that $c_\lambda(x, y) \leqq \min(x, y)$, we have

$$\min(x, y) \geqq \lim_{\lambda\to\infty} c_\lambda(x, y) \geqq \lim_{\lambda\to\infty} f_c^{(-1)}((2f_c^{\lambda}(x))^{1/\lambda}) = \lim_{\lambda\to\infty} f_c^{(-1)}(2^{1/\lambda}(f_c(x))) =$$
$$= x = \min(x, y).$$

2. If $y=1$, then $c_\lambda(x, 1)=x$. Similarly, if $x=1$, then $c(1, y)=y$.
Let us assume that $x<1$, $y<1$ and $x \leqq y$. Since

$$0 \leqq \lim_{\lambda\to 0} c_\lambda(x, y) = \lim_{\lambda\to 0} f_c^{(-1)}((f_c^{\lambda}(x)+f_c^{\lambda}(y))^{1/\lambda}) = \lim_{\lambda\to 0} f_c^{(-1)}(2^{1/\lambda} f_c(y)) = 0,$$

we have

$$c_0(x, y) = t_c(x, y), \qquad d_0(x, y) = t_d(x, y),$$
$$c_\infty(x, y) = \min(x, y), \qquad d_\infty(x, y) = \max(x, y).$$

Example. Let $f_c(x)=1-x$. We then obtain the operator of Yager [10] by constructing $c(x, y)$.

Definition. $c(x, y)\,(d(x, y))$ has the classification property, if $c(x, n(x))=0$ $(d(x, n(n))=1)$ for every negation $n(x)$.

Theorem 11. Let $c(x, y)\,(d(x, y))$ be any mapping such that it is isotonic, and satisfies $c(1, x)=c(x, 1)=x$ $(d(0, x)=d(x, 0)=x)$.

(i) $c(x, y)\,(d(x, y))$ is idempotent if and only if

$$c(x, y)=\min(x, y)\ (d(x, y)=\max(x, y)),$$

(ii) $c(x, y)\,(d(x, y))$ has the classification property if and only if

$$c(x, y) = t_c(x, y), \quad d(x, y) = t_d(x, y).$$

Proof. The sufficiency is obvious.

(i) Let us assume that $c(x, y)$ is idempotent and $x \leqq y$. Then

$$x = c(x, x) \leqq c(x, y) \leqq c(x, 1) = x \quad \text{i.e.}$$

$$c(x, y) = x = \min(x, y).$$

(ii) If $x=1$ or $y=1$, then the statement is obvious. If $x_0<1$, $y_0<1$, then there is a negation $n(x)$ for which $x_0=n(y_0)$. Since $c(x, y)$ has the classification property we have $c(x_0, y_0)=c(x_0, n(x_0))=t_c(x_0, y_0)$.

Hence, we have characterized the operators $c_0(x, y)$, $c_\infty(x, y)$ algebraically, too.

5. The distributive operator class

We wish to describe operators satisfying properties (i)—(iv) which are distributive with respect to each other, i.e.

$$c(x, d(y, z)) = d(c(x, y), c(x, z)), \tag{22}$$

$$d(x, c(y, z)) = c(d(x, y), d(x, z)). \tag{23}$$

Theorem 12. The operators $c(x, y)$ and $d(x, y)$ are distributive with respect to each other if and only if:

$$c(x, y) = \min(x, y), \quad d(x, y) = \max(x, y). \tag{24}$$

Proof. On the basis of theorem 1, boundary conditions (6) and (7) hold. Utilizing the distributivity:

$$x=d(x, 0)=d(x, c(0, 0))=c(d(x, 0),\ d(x, 0))=c(x, x)$$

we have that is $c(x, y)$ is idempotent. On the basis of theorem 11, for an idempotent, isotonic conjunctive (disjunctive) operator satisfying the boundary conditions

$$c(x, y) = \min(x, y).$$

6. Discussion

The properties of monotonous fuzzy operators (interpreted on sets) satisfying associativity have been examined.

A_1 Strictly monotonous operators,
A_2 Archimedean operators,
A_3 Monotonous operators

A necessary and sufficient condition on generator functions was formulated which ensures that De Morgan identity is true. The distributivity holds if and only

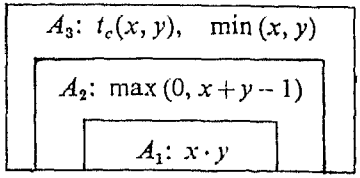


Fig. 3
Hierarchic order of operators

if the operator is the min or max. By means of operator series associated to the operators we obtained lower and upper limits of the operators, as limes. The algebraic characterization of these operators was also given.

A further examination is necessary to establish the classification property of the operators and the relation of the (not strictly monotonous) negations.

References

[1] ACZÉL, J. Lectures on functional equations and their applications, Academic Press, New York (1966).
[2] ALBERT, P., The algebra of fuzzy logic, Fuzzy Sets and Systems 1. 203—230, 1978.
[3] DOMBI, J., A general class of fuzzy operators, the De Morgan class of fuzzy operators Fuzzy Sets and Systems 8. 149—163, (1982).
[4] DOMBI, J., and VAS, Z., Basic theoretical treatment of fuzzy connectives Acta Cybernet. v. VI. 2, (1983), pp. 191—201.
[5] DREWNIAK, J., Axiomatic systems in fuzzy algebra, Acta Cybernet. v. V. 2, (1981), pp. 191—206.
[6] DUBOIS, D. and PRADE, H., New results about properties and semantics of fuzzy set theoretic operators, 1-st Symposium on Policy Analysis and Information Systems. Durham, North Carolina, USA (1979).
[7] FUNG, L. W. and FU, K. S., An axiomatic approach to rational decision making in fuzzy environment in fuzzy sets and their applications to cognitive and decision processes, ed. by Zadeh, L. A. and Fu K. S. and Tanaka, K. and Shimura M., Academic Press, 227—256, (1975).
[8] HAMACHER, H., Über logische Verknupfungen unscharfer Aussagen und daren zugeharige Bewertungs-functionene, Progress in Cybernetics and Systems Research Vol. III. ed. by Trappl, R. and Klir, G. J. and Riccardi, L., John Wiley and Sons, New York, 276—288, (1979).
[9] LING, C. H., Representations of associative functions, Publ. Math. Debrecen, 12, 182—212, (1965).
[10] YAGER, R. R., On a general class of fuzzy connectives, Fuzzy Sets and Systems 4, 235—242, (1980).

(Received Nov. 17, 1984)

On the optimization of library information retrieval systems

By A. M. IVÁNYI and A. N. SOTNIKOW

Introduction

In some information retrieval systems (e.g. in many library ones) the content of documents (books, papers, patents, computer programs etc.) and the content of user's questions are characterized by using a given set of key-words (so called descriptors). The aim of the retrieval is to find all the documents whose content is characterized by the descriptors in the question and, may be, by other descriptors too.

Of course, the sequential retrieval represents a slow method, which is satisfactory only in small systems.

A more effective method is proposed by G. Salton [1]. According to this method the set of documents is decomposed into disjoint subsets (so called clusters) consisting of "similar" documents, and the retrieval is organized in a hierarchical manner. Recently an excellent review was publicated by S. T. March on the different retrieval methods [2].

According to Salton's proposition [3] the efficiency of the retrieval system is characterized by the expected waiting time of the users, therefore the aim of the optimization is to minimize this time choosing suitable parameters (number of clusters, sizes of clusters, distribution of documents among the clusters).

At first in § 1 we describe a mathematical model of such systems [1], then in § 2 the mentioned expected time is computed for some values of parameters, later in § 3 this time is analysed as a function of the different parameters, and finally in § 4 some practical conclusions are made.

§ 1. The mathematical model

Let m, n, r and z be positive integers, $\delta=\{D_0, \ldots, D_i, \ldots, D_{n-1}\}$ — documents, $\varkappa=\{K_0, \ldots, K_j, \ldots, K_{m-1}\}$ — key-words, $\varrho=\{Q_0, \ldots, Q_p, \ldots, Q_{r-1}\}$ — possible questions of users, $\alpha=\{A_0, \ldots, A_p, \ldots, A_{r-1}\}$ — the corresponding answers of the information retrieval system, where $Q_p \subseteq \varkappa$, $A_p \subseteq \delta$ for $p=0, \ldots, r-1$.

The content of documents is characterized by the document description matrix $\mathbf{D}=[d_{ij}]_{n\times m}$, where $d_{ij}=1$, if K_j characterizes D_i, and $d_{ij}=0$ otherwise. The content of questions is characterized by using the question description matrix $\mathbf{Q}=[q_{pj}]_{r\times m}$,

where $q_{pj}=1$, if K_j characterizes Q_p, and $q_{pj}=0$ otherwise. The i-th row $\mathbf{d}_i$ of $\mathbf{D}$ represents the description of D_i, and the p-th row $\mathbf{q}_p$ of Q represents the description of Q_p. In the case of the question Q_p we have to find the documents D_i, for which $\mathbf{d}_i \geqq \mathbf{q}_p$.

The set δ is decomposed into disjoint clusters $C_0, \ldots, C_k, \ldots, C_{z-1}$. The decomposition is defined using the decomposition matrix $\mathbf{Y}=[y_{ik}]_{n\times z}$, where $y_{ik}=1$, if $D_i \in C_k$ and $y_{ik}=0$ otherwise. The content of clusters is described by using the characteristic matrix $\mathbf{U}=[u_{kj}]_{z\times m}$, where $u_{kj}=0$ if for any $D_i \in C_k$, $d_{ij}=0$, and $u_{kj}=1$ otherwise. The k-th row $\mathbf{u}_k$ of $\mathbf{U}$ is called the characteristic vector of C_k.

The retrieval consists of two levels: on the first level the description $\mathbf{q}_p$ of the question Q_p is compared with the characteristic vectors $\mathbf{u}_k$ for $k=0, \ldots, z-1$. On the second level a sequential retrieval is realized in the relevant clusters: a cluster C_r is relevant to Q_p iff $\mathbf{u}_k \geqq \mathbf{q}_p$ ($u_{k,j} \geqq q_{p,j}$ for $j=0, 1, \ldots, m-1$).

As a time unit (and as a cost unit, too) let us choose the time required to compare two m-dimensional vectors. Then in the case of a decomposition matrix $\mathbf{Y}$ the cost $S_{p\mathbf{Y}}$ of the answer A_p to a question Q_p is defined by

$$S_{p\mathbf{Y}} = z + \sum_{k=0}^{z-1} |C_k| L_{kp} + t \sum_{k=0}^{z-1} L_{kp},$$

where $L_{kp}=1$, if C_k is relevant to Q_p and $L_{kp}=0$ otherwise, t is a nonnegative real parameter (the time, required to prepare a new cluster to the processing). The efficiency of a given decomposition algorithm A, resulting a decomposition matrix $\mathbf{Y}_A$ is denoted by $\mathbf{M}(\mathbf{Y}_A)$ and defined by the average cost of answers to all possible questions

$$\mathbf{M}(\mathbf{Y}_A) = \frac{1}{r} \sum_{p=0}^{r-1} S_{p\mathbf{Y}}.$$

For example, let $m=2$, $n=r=4$, $z=2$, $t=0$,

$$\mathbf{D} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Then the corresponding answers are $A_0=\{D_0, D_1, D_2, D_3\}$, $A_1=\{D_1, D_3\}$, $A_2=\{D_2, D_3\}$ and $A_3=\{D_3\}$.

Let us assume that we construct two different decompositions defined by the matrices

$$\mathbf{Y}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{Y}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Then the corresponding characteristic matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ are as follows:

$$\mathbf{U}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{U}_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

If all the four possible questions are put once, then we have eight comparisons on the first level for both decompositions; the second decomposition requires $4\times4=16$ ones on the second level, while the first needs only $2\times4+2\times2=12$, therefore, $\mathbf{M}(\mathbf{Y}_1)=\frac{20}{4}=5$, while $\mathbf{M}(\mathbf{Y}_2)=\frac{24}{4}=6$.

§ 2. Efficiency of a decomposition algorithm

Let m be fixed, $n=r=2^m$. Let us suppose, that the descriptions of the documents (the rows of the matrix $\mathbf{D}$) and the descriptions of the questions (the rows of the matrix $\mathbf{Q}$) are different, that is $0\leqq i<I\leqq n-1$ implies $\mathbf{d}_i\neq\mathbf{d}_I$, and $0\leqq p<P\leqq r-1$ implies $\mathbf{q}_p\neq\mathbf{q}_P$. In this case we have 2^m possible different documents and also 2^m different questions. For the simplicity let us suppose, that the descriptions as binary numbers give the corresponding indices, that is

$$i=\sum_{j=0}^{m-1} d_{ij}2^{m-1-j}, \quad p=\sum_{j=0}^{m-1} q_{pj}2^{m-1-j} \quad (i, p=0, \ldots, 2^m-1).$$

Let us suppose that x is a nonnegative integer and $|C_k|=2^x$ for $k=0, 1, \ldots, z-1$. In this case we have $z=2^{m-x}$ and $0\leqq x\leqq m$.

For this special values of parameters the decomposition algorithm S [4] is defined as follows:

$$C_k=\{D_{k\cdot 2^x}, D_{k\cdot 2^x+1}, \ldots, D_{k\cdot 2^x+2^x-1}\} \quad (k=0, 1, \ldots, 2^{m-x}-1).$$

Now we can give the efficiency of S as a function of parameters x, m and t [4].

Theorem 1. If x, m are positive fixed integers, $n=r=2^m$, t is a nonnegative real number, $0\leqq i<I<n$ implies $\mathbf{d}_i\neq\mathbf{d}_I$, $0\leqq p<P<r$ implies $\mathbf{q}_p\neq\mathbf{q}_P$, then

$$\mathbf{M}(\mathbf{Y}_S)=2^{m-x}+2^{2x-m}3^{m-x}+t\cdot 2^{x-m}3^{m-x}. \quad \square \tag{1}$$

Proof. According to the definition $\mathbf{M}(\mathbf{Y}_S)$ we have

$$\mathbf{M}(\mathbf{Y}_S)=\frac{1}{2^m}\left[2^{2m-x}+2^x\sum_{k=0}^{z-1}\sum_{\mathbf{q}_p\leqq\mathbf{u}_k}1+t\sum_{k=0}^{z-1}\sum_{\mathbf{q}_p\leqq\mathbf{u}_k}1\right].$$

In this expression the sum $\sum_{\mathbf{q}_p\leqq\mathbf{u}_k}1$ is equal to the number of questions for which $(q_{p,0}, \ldots, q_{p,m-1})\leqq(u_{k,0}, \ldots, u_{k,m-1})$.

Considering $\mathbf{u}_k$ as a binary number b_k according to the grouping rule of S we get

$$b_k=\sum_{j=0}^{m-1} u_{k,j}2^{m-j-1}=(k+1)2^x-1$$

(the first $m-x$ digits determine k, the last x digits are 1's).

If $\mathbf{u}_k$ contains w 1's on the first $m-x$ places, then we have $2^w\cdot 2^x$ questions with $\mathbf{q}_p\leqq\mathbf{u}_k$. Therefore, using the equality

$$\sum_{w=0}^{m-x}\binom{m-x}{w}2^w=(2+1)^{m-x}$$

we get

$$\mathbf{M}(\mathbf{Y}_S) = 2^{m-x} + (2^{x-m} + t \cdot 2^{-m}) \sum_{w=0}^{m-x} \binom{m-x}{w} 2^w 2^x =$$

$$= 2^{m-x} + 2^{2x-m} 3^{m-x} + t \cdot 2^{x-m} 3^{m-x}. \quad \square$$

§ 3. Analysis of the influence of the continuous parameters

According to the Theorem 1 the cost $\mathbf{M}(\mathbf{Y}_S)$ depends on x, m and t as

$$\mathbf{M} = \mathbf{M}(x, t, m) = e^{mb} e^{-xb} + e^{m(c-b)} e^{x(2b-c)} + t \cdot e^{m(c-b)} e^{-x(c-b)},$$

where $b = \ln 2 \approx 0{,}69314718$ and $c = \ln 3 \approx 1{,}09861229$, t is a nonnegative real parameter, m is a positive integer parameter, and x represents the independent variable being integer and $0 \leqq x \leqq m$.

In the practice the number of key-words m equals 10—1000 [4, 5], the time of preparation of a new cluster to the processing t is about 0—10^9 time units (may be, it is necessary to put new disc or magnetic tape).

In the analysis of the function $\mathbf{M} = \mathbf{M}(x, t, m)$ its continuous variant $\mathbf{R} = \mathbf{R}(x, t, m)$, plays an important role, where x, t and m are real variables with $x \in (-\infty, +\infty)$, $t \in (-\infty, +\infty)$, $m \in (-\infty, +\infty)$.

Lemma 1. If $t \geqq 0$ and m are fixed real numbers, then the function $\mathbf{R} = \mathbf{R}(x, t, m)$ as a function of x is convex, it has one local minimum and this minimum represents an absolute one too. $\square$

Proof. Differentiating $\mathbf{R}(x, t, m)$ by x we get

$$\mathbf{R}_x = -b \cdot e^{mb} e^{-bx} + (2b - c) e^{m(c-b)} e^{x(2b-c)} - (c - b) t \cdot e^{m(c-b)} e^{-x(c-b)}.$$

Since $\mathbf{R}_x$ is a monotone increasing continuous function of x (its members are increasing) and its limit is $+\infty$ for $x \to +\infty$ and $-\infty$ for $x \to -\infty$, so $\mathbf{R}_x$ has a unique zero-place x_0 and $\mathbf{R}_x < 0$ for $x < x_0$, $\mathbf{R}_x > 0$ for $x > x_0$. Therefore $\mathbf{R}(x, t, m)$ is convex, and it has an absolute minimum at x_0. $\square$

Lemma 2. For a fixed m the function $x_0 = x_0(t, m)$ is a monotone increasing function of t. $\square$

Proof. From $\mathbf{R}_x = 0$ we get

$$t = t(x_0, m) = \frac{2b - c}{c - b} e^{bx_0} - \frac{b}{c - b} e^{m(2b-c)} e^{-x_0(2b-c)}.$$

For a fixed m the function $t(x_0, m)$ is a monotone increasing one of x_0 for $x_0 \in (-\infty, +\infty)$, so it has a monotone increasing inverse $x_0 = x_0(t, m)$. $\square$

Lemma 3. For fixed $t \geqq 0$ the function $x_0 = x_0(t, m)$ is a monotone increasing function of m. $\square$

Proof. It follows from $\mathbf{R}_x=0$ that

$$m=\frac{1}{2b-c}\ln\left[\frac{2b-c}{b}e^{x_0(3b-c)}-t\frac{c-b}{b}e^{x_0(2b-c)}\right].$$

According to Lemma 1 for given t and m there exists a corresponding x_0, therefore the expression in the square brackets has to be positive (otherwise m has no meaning).

Now we have

$$\frac{\partial m}{\partial x_0}=\frac{\frac{2b-c}{b}(3b-c)e^{(3b-c)x_0}-\frac{t(c-b)(2b-c)}{b}e^{x_0(2b-c)}}{(2b-c)\left[\frac{2b-c}{b}e^{x_0(3b-c)}-\frac{t(c-b)}{b}e^{x_0(2b-c)}\right]}.$$

The expression in the numerator is greater than the one in the square brackets of the denominator (the positive member is multiplied by $3b-c$, the negative one by $2b-c$), therefore $\frac{\partial m}{\partial x_0}>0$ and so the derivative $\frac{\partial x_0}{\partial m}$ also is everywhere positive and x_0 is an increasing function of m. □

Theorem 2. If $t\geqq 0$ and m are fixed real numbers, then

$$x_0\begin{cases}=\beta m+\gamma, & \text{if } t=0,\\ >\beta m+\gamma, & \text{if } t>0,\end{cases}$$

where $\beta=\frac{\ln 4/3}{\ln 8/3}\approx 0{,}29330495$ and $\gamma=\frac{\ln\frac{\ln 2}{\ln 4/3}}{\ln 8/3}\approx 0{,}89657440$. □

Proof. Lemma 1 and the equation $\mathbf{R}_x=0$ at the condition $t=0$ imply $x_0=\beta m+\gamma$, and then Lemma 2 gives the remaining part of the theorem. □

§ 4. Conclusions

In the previous paragraph we considered x, t and m as continuous parameters. In the practice x and m have positive integer values.

According to Lemma 1, $\mathbf{R}(x, t, m)$ as a function of x is a convex function, therefore Theorem 2 implies for the optimal size-parameter x_{opt} the assertion: if $t=0$, then

$$\lfloor\beta m+\gamma\rfloor\leqq x_{opt}\leqq\lceil\beta m+\gamma\rceil.$$

If $t>0$ then we have a lower bound $x_{opt}\geqq\lfloor\beta m+\gamma\rfloor$ due to Lemma 2.

Analysing the function $\mathbf{M}(x, t, m)$ one can see that for small values of t the third member of (1) has a relatively weak influence on the value of x_{opt}, and so in the first approximation is neglectable.

We propose the following approximate formula:

$$x_{opt}\approx\lfloor 0{,}29330495m+0{,}89657440\rfloor. \tag{2}$$

Table 1. Summary of the numerical results

m	t	x_{opt}	x_a	m	t	x_{opt}	x_a
20	10^0	7	7	80	10^0	24	24
	10^2	8	7		10^2	24	24
	10^4	14	7		10^4	24	24
	10^6	20	7		10^6	24	24
40	10^0	13	13	100	10^0	30	30
	10^2	13	13		10^2	30	30
	10^4	14	13		10^4	30	30
	10^6	20	13		10^6	30	30
60	10^0	19	19				
	10^2	19	19				
	10^4	19	19				
	10^6	21	19				

The following table shows some numerical results of a BASIC-program, running on a personal computer Commodore-64.

In this table x_{opt} represents the optimal integer value of x and x_a is the value resulted by the approximate formula (2).

According to the presented numerical data formula (2) results good approximations of the optimal size parameter for $m>20$ and $t<1\,000\,000$.

The authors are indebted to Dr. Péter Simon at Eötvös Loránd University of Budapest for the consultation and valuable remarks.

References

[1] Salton, G., Automatic information organization and retrieval. McGraw Hill, New York, 1968.
[2] March, S. T., Techniques for structuring databese records. Computing Surveys, *15* (1) (1983), 45—79.
[3] Salton, G., Dynamic information and library processing. Prentice Hall Inc., Englewood Cliffs, 1975.
[4] Ивани А. М. и Сотников А. Н. Об оптимизации дескрипторных АИПС с зонно-иерархической организацией поискового множества. Вестник Московского Университета, Сер. 15. Вычислительная Математика и Кибернетика, (2) (1984) 53—57.
[5] Iványi, A. M. and Nyirádi, L., Library retrieval systems INF and FARMDOC. In: Proc. of Conf. for information-servicing of professionally linked computer users (Varna, 1977), 307—313.

(Received March 13, 1985)

A probability model for priority processor-shared multiprogrammed computer systems

By J. SZTRIK

1. Introduction

Queueing models have been widely used in the analysis of time-shared computer systems. In these systems an arriving job competes for the attention of the single CPU. It is forced to wait in a system of queues until it is permitted a quantum Q of service time. When this quantum expires, it is then required to join the system of queues to await its required service time. By allowing Q to shrink to zero, processor-sharing (PS) models are obtained, which provides a share of the CPU to many jobs simultaneously and equally.

Following Kleinrock [6] in a priority round-robin system the jobs are divided into n separate priority groups. A program belonging to the i-th priority group gets r_iQ unit of service each time, where quantities r_i are called service weights, $r_i>0$, $i=1, \ldots, n$. In the limit as $Q\to 0$ this model reduces to a processor-shared one with priority structure wherein a job from priority group i receives a fraction f_i of the total capacity, where $f_i=r_i/\Sigma r_j n_j$, here n_j is the number of jobs from group j in the system at time t. This kind of service discipline is referred to as PPS one. The PS model is a particular case $r_i=1$ for all i, $i=1, \ldots, n$.

We observe that the two processor-shared models are ideal in the sence that the swap-time is assumed to be zero.

The present paper deals with a multiprogrammed computer system in which a number of n jobs are permitted to circulate among the peripheral devices and the CPU. The system is assumed to have enough peripheral devices, so no queueing for I/0 operations occurs. Under PPS service discipline the jobs are supposed to be stochastically different, the i-th program is characterized by exponentially distributed I/0 time with parameter λ_i, exponentially distributed processing time with rate μ_i and service weight r_i, $i=1, \ldots, n$. All random variables are assumed to be independent of each other.

The purpose of the paper is to generalize the PS model treated by Asztalos [1], Csige—Tomkó [4], Cohen [3]. In steady state the main operational characteristics, such as CPU utilization, expected CPU busy period length, mean response times, waiting ratio, throughput, average number of jobs staying at the CPU are given.

Furthermore, a system of linear equations for L—S transform of response time

for program i and CPU busy period length is obtained, respectively, which can be solved by the algorithm offered. For the moments of the random variables mentioned before another system of linear equations is derived which can be solved iterative.

Finally, numerical examples illustrate the problem in question and performance measures under different service disciplines, such as preemptive resume priority (PR), PS are compared with the PPS one.

For further probabilistic models for multiprogrammed computer systems the interested reader is referred to among others: Avi—Itzhak and Heyman [2], Cohen [3], Kleinrock [5], Lehoczky and Gaver [7], Sztrik [8].

The theoretical basis of the paper can be found in Tomkó [9].

2. Mathematical description of the model

Let the random variable $v(t)$ denote the number of jobs at the CPU at time t, and let $(\alpha_1(t), \ldots, \alpha_{v(t)}(t))$ indicate their indexes ordered lexicographically. Introduce the process

$$\underline{x}(t) = (v(t); \alpha_1(t), \ldots, \alpha_{v(t)}(t)).$$

Since all distributions are exponential the process $(\underline{x}(t), t \geqq 0)$ is a stochastically continuous, finite state, continuous time Markov chain with state space $\bigcup_{k=1}^{n} C_k^n + \{0\}$, where C_k^n denotes the set of all combinations of order k of the integers $1, \ldots, n$ in lexicograpic order and $\{0\}$ indicates the state that the CPU is idle.

Let us introduce some notations

$$\Lambda_{i_1, \ldots, i_k} = \sum_{j \neq i_1, \ldots, i_k} \lambda_j, \quad \Lambda = \sum_{j=1}^{n} \lambda_j,$$

$$R_{i_1, \ldots, i_k} = \sum_{j=1}^{k} r_{i_j}, \quad \sigma_{i_1, \ldots, i_k} = \frac{1}{R_{i_1, \ldots, i_k}} \sum_{j=1}^{k} r_{i_j} \mu_{i_j} + \Lambda_{i_1, \ldots, i_k}.$$

For the distribution of $\underline{x}(t)$ consider the functions given below

$$P_0(t) = P(v(t) = 0),$$

$$P_{i_1, \ldots, i_k}(t) = P(v(t) = k;\ \alpha_1(t) = i_1, \ldots, \alpha_k(t) = i_k), \tag{1}$$

$$(1 \leqq k \leqq n, (i_1, \ldots, i_k) \in C_k^n).$$

It is easy to see that functions (1) satisfy the Kolmogorov-differential equations

$$P_0'(t) = -\Lambda P_0(t) + \sum_{j=1}^{n} \mu_j P_j(t),$$

$$\vdots$$

$$P_{i_1, \ldots, i_k}'(t) = \sum_{l=1}^{k} \lambda_{i_l} P_{i_1, \ldots, i_{l-1}, i_{l+1}, \ldots, i_k}(t) - \sigma_{i_1, \ldots, i_k} P_{i_1, \ldots, i_k}(t) + \tag{2}$$

$$+ \sum_{j \neq i_1, \ldots i_k} \frac{\mu_j r_j}{R_{i_1, \ldots, i_k, j}} P_{i_1', \ldots, i_{k+1}'}(t),$$

$$\vdots$$

$$P_{1,2,\ldots,n}'(t) = \sum_{l=1}^{n} \lambda_l P_{1, \ldots, l-1, l+1, \ldots, n}(t) - \sigma_{1, \ldots, n} P_{1, \ldots, n}(t),$$

where $i'_1, \ldots, i'_{k+1}$ denotes the lexicographic order of integers $i_1, \ldots, i_k, j$. Then we have:

Theorem 1. If $\lambda_i, \mu_i > 0$, $i=1, \ldots, n$ then the Markov chain $(\underline{x}(t), t \geqq 0)$ possesses unique stationary distribution

$$P_0 = \lim_{t \to \infty} P_0(t),$$

$$P_{i_1, \ldots, i_k} = \lim_{t \to \infty} P_{i_1, \ldots, i_k}(t),$$

$$(i_1, \ldots, i_k) \in C_k^n, \quad k = 1, \ldots, n,$$

which is the solution to the following system of linear equations

$$\begin{aligned}
\Lambda P_0 &= \sum_{j=1}^{n} \mu_j P_j, \\
&\vdots \\
\sigma_{i_1, \ldots, i_k} P_{i_1, \ldots, i_k} &= \sum_{l=1}^{k} \lambda_{i_l} P_{i_1, \ldots, i_{l-1}, i_{l+1}, \ldots, i_k} + \sum_{j \neq i_1, \ldots, i_k} \frac{\mu_j r_j}{R_{i_1, \ldots, i_k, j}} P_{i'_1, \ldots, i'_{k+1}}, \qquad (3) \\
&\vdots \\
\sigma_{1, \ldots, n} P_{1, \ldots, n} &= \sum_{l=1}^{n} \lambda_l P_{1, \ldots, l-1, l+1, \ldots, n},
\end{aligned}$$

satisfying the norming condition

$$P_0 + \sum_{k=1}^{n} \sum_{c_k^n} P_{i_1, \ldots, i_k} = 1. \qquad (4)$$

Proof. Since $(\underline{x}(t), t \geqq 0)$ is a continuous time finite state Markov chain with positive intensities, it is irreducible and all states are ergodic. In this case the stationary distribution exists and can be obtained as the solution to the Kolmogorov equations satisfying (4). As $t \to \infty$ from (2) we get (3).

If all $r_i = 1$, $i=1, \ldots, n$, the solution of (3) is

$$P_{i_1, \ldots, i_k} = P_0 K! \prod_{j=1}^{k} \lambda_{ij} / \mu_{ij},$$

(cf. Csige—Tomko [4], Asztalos [1]).

In the following we give an algorithm for calculating the stationary distribution $(P_0, P_{i_1}, \ldots, i_k, (i_1, \ldots, i_k) \in C_k^n, k=1, \ldots, n)$.

Let $\underline{Y}_k$ be the vector

$$\begin{pmatrix} P_{1, \ldots, k} \\ \vdots \\ P_{i_1, \ldots, i_k} \\ \vdots \\ P_{n-k+1, \ldots, n} \end{pmatrix}$$

of dimension $\binom{n}{k}$. The components $P_{i_1, \ldots, i_k}$ are listed in the lexicographic order of their indexes, $k=1, \ldots, n$. Notice that eq. (3) can be written in the following

neat form

$$\underline{Y}_0 = \mathbf{B}_0\underline{Y}_1,$$

$$\underline{Y}_1 = \mathbf{A}_1\underline{Y}_0+\mathbf{B}_1\underline{Y}_2,$$

$$\vdots$$

$$\underline{Y}_k = \mathbf{A}_k\underline{Y}_{k-1}+\mathbf{B}_k\underline{Y}_{k+1},$$

$$\vdots$$

$$\underline{Y}_n = \mathbf{A}_n\underline{Y}_{n-1},$$

where matrix $\mathbf{A}_k$ is of order $\binom{n}{k}\times\binom{n}{k-1}$ $k=1, \ldots, n$, $\mathbf{B}_k$ is of order $\binom{n}{k}\times\binom{n}{k+1}$ $k=0, 1, \ldots, n-1$, $Y_0=P_0$. The elements of $\mathbf{A}_k$, $\mathbf{B}_k$ can be determined by the help of eq. (3). The solution to (5) can be obtained by an iterative manner $\underline{Y}_k=\mathbf{F}_k\underline{Y}_{k-1}$, $k=1, \ldots, n$. To verify this let $\mathbf{F}_n=\mathbf{A}_n$, furthermore assume that $\underline{Y}_{k+1}=\mathbf{F}_k\underline{Y}_k$. Let us consider equation $\underline{Y}_k=\mathbf{A}_k\underline{Y}_{k-1}+\mathbf{B}_k\underline{Y}_{k+1}$. After substituting we get $(1-\mathbf{B}_k\mathbf{F}_{k+1})\underline{Y}_k=\mathbf{A}_k\underline{Y}_{k-1}$ then

$$\underline{Y}_k = (1-\mathbf{B}_k\mathbf{F}_{k+1})^{-1}\mathbf{A}_k\underline{Y}_{k-1}.$$

Let

$$\mathbf{F}_k = (1-\mathbf{B}_k\mathbf{F}_{k+1})^{-1}\mathbf{A}_k,$$

so $\underline{Y}_k=\mathbf{F}_k\underline{Y}_{k-1}$, $k=1, \ldots, n$. Starting with any Y_0 after norming the stationary distribution is given.

3. Performance measures

In the following $(\underline{x}(t), t\geqq 0)$ is supposed to be in equilibrium.

(i) *CPU utilization.* Using renewal-theoretic arguments it is well-known that

$$P_0 = (1/\Lambda)(1/\Lambda+M\delta)^{-1},$$

where $M\delta$ denotes the mean CPU busy period length and $1/\Lambda$ is its average idle period length. If the CPU utilization, which is the long-run fraction of time the CPU is busy, is denoted by U we have

$$U = 1-P_0 = (M\delta)(1/\Lambda+M\delta)^{-1}.$$

Consequently,

$$M\delta = (1-P_0)/\Lambda P_0.$$

(ii) *Mean response times.* During the execution a job is served by the CPU and takes I/0 operations. If these periods are considered as cycles, then in steady state these cycles lengths are identically distributed but not independent random variables.

Let $P^{(i)}$ denote the stationary probability that job i is in compute period and let the average response time be designated by R_i. Furthermore, let H_i be the event that the program i is under service. Introduce the function

$$Z_{H_i}(t) = \begin{cases} 1 & \text{if } \underline{x}(t)\in H_i, \\ 0 & \text{otherwise.} \end{cases}$$

By the virtue of Theorem 1 and 3 in Tomkó [9] we have

$$\lim_{T\to\infty}\frac{1}{T}\int_0^T Z_{H_i}(t)\,dt = P^{(i)} = \frac{R_i}{1/\lambda_i + R_i}.$$

Since $P^{(i)}$ can be easily evaluated as

$$P^{(i)} = \sum_{k=1}^{n}\sum_{i\in(i_1,\ldots,i_k)\in c_k^n} P_{i_1,\ldots,i_k},$$

for the expected response time of job i we obtain

$$R_i = P^{(i)}/\lambda_i(1-P^{(i)}).$$

It is clear that $\Sigma Z_{H_i}(t)$ gives the number of jobs statying at the CPU at time t. Thus in equilibrium the mean number of programs processed by the CPU is $\bar{n}=\Sigma P^{(i)}$. In addition, the Little's formula is valid

$$\Sigma\lambda_i(1-P^{(i)})R_i = \Sigma P^{(i)} = \bar{n}.$$

(iii) *Waiting ratio.* Defining the waiting ratio for job i by $\hat{W}_i=\mu_i(R_i-1/\mu_i)$ the system waiting ratio can be obtained as

$$\hat{W} = \Sigma\hat{W}_i = \Sigma\mu_i R_i - n.$$

(iv) *Throughput.* Denote by T the throughput of the system, which is the mean number of jobs serviced in unit time. If T_i denotes the throughput for job i, we have

$$T_i = \lambda_i(1-P^{(i)}).$$

Thus, we get

$$T = \Sigma T_i = \Sigma\lambda_i(1-P^{(i)}).$$

4. L—S transform of the CPU busy period and response times

Let us denote by η_α a random variable distributed exponentially with rate α. If η_α and η_β are independent, then $\min(\eta_\alpha, \eta_\beta)=\eta_{\alpha+\beta}$ which is a well-known fact. Furthermore, let the notation $(\eta_{\alpha_i}=\eta_{\alpha_1+\ldots+\alpha_s})$ mean the event that $\min(\eta_{\alpha_1}, \ldots, \eta_{\alpha_s})=\eta_{\alpha_i}$, probability of which is $\alpha_i/\sum_{j=1}^{s}\alpha_j$, where $\alpha_i>0$, $i=1, \ldots, s$.

Let $\chi(A)$ denote the characteristic function of event A, i.e.

$$\chi(A) = \begin{cases} 1 & \text{if } A \text{ occurs,} \\ 0 & \text{otherwise.} \end{cases}$$

(i) *CPU busy period length.* Let the random variable $\delta_{i1}, \ldots, i_k$ denote a busy time interval of the CPU initiated by state $(i_1, \ldots, i_k)$, $(i_1, \ldots, i_k)\in C_k^n$, $k=1, \ldots, n$.

Similarly to Csige—Tomkó [4] the following recursive relations hold

$$\begin{aligned}
\delta_i &\doteq \eta\sigma_i + \sum_{l\neq i} \delta_{i',l'}\chi(\eta_{\lambda_l} = \eta\sigma_i),\\
&\vdots\\
\delta_{i_1,\ldots,i_k} &\doteq \eta\sigma_{i_1,\ldots,i_k} + \sum_{j=1}^{k} \delta_{i_1,\ldots,i_{j-1},i_{j+1},\ldots i_k}\chi\left(\eta\frac{\mu_{ij}r_{ij}}{R_{i_1,\ldots,i_k}} = \eta\sigma_{i_1,\ldots,i_k}\right) +\\
&\quad + \sum_{l\neq i_1,\ldots,i_k} \delta_{i'_1,\ldots,i'_{k+1}}\chi(\eta_{\lambda_l} = \eta\sigma_{i_1,\ldots,i_k}),\\
&\vdots\\
\delta_{1,\ldots,n} &\doteq \eta\sigma_{1,\ldots,n} + \sum_{j=1}^{n} \delta_{1,\ldots,j-1,j+1,\ldots,n}\chi\left(\eta\frac{\mu_j r_j}{R_{1,\ldots,n}} = \eta\sigma_{1,\ldots,n}\right)
\end{aligned} \tag{6}$$

where $\doteq$ denotes that the equality is meant in distribution. Introducing the L—S transform

$$g_{i_1,\ldots,i_k}(s) = Me^{-s\delta_{i_1,\ldots,i_k}},$$

from (6) we get

$$\begin{aligned}
g_i(s) &= \frac{1}{s+\sigma_i}\left[\mu_i + \sum_{l\neq i}\lambda_l g_{i',l'}(s)\right],\\
&\vdots\\
g_{i_1,\ldots,i_k}(s) &= \frac{1}{s+\sigma_{i_1,\ldots,i_k}}\left[\sum_{j=1}^{k}\frac{\mu_{ij}r_{ij}}{R_{i_1,\ldots,i_k}} g_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_k}(s) +\right.\\
&\quad \left. + \sum_{l\neq i_1,\ldots,i_k}\lambda_l g_{i'_1,\ldots,i'_{k+1}}(s)\right],\\
g_{1,\ldots,n}(s) &= \frac{1}{s+\sigma_{1,\ldots,n}}\sum_{j=1}^{n}\frac{\mu_j r_j}{R_{1,\ldots,n}} g_{1,\ldots,j-1,j+1,\ldots,n}(s).
\end{aligned} \tag{7}$$

Finally,

$$Me^{-s\delta} = \sum_{i=1}^{n}\frac{\lambda_i}{\Lambda} g_i(s).$$

Let $\underline{G}_k(s)$ be the vector

$$\begin{vmatrix} g_{1,\ldots,k}(s)\\ \vdots\\ g_{i_1,\ldots,i_k}(s)\\ \vdots\\ g_{n-k+1,\ldots,n}(s)\end{vmatrix}$$

of dimension $\binom{n}{k}$. The components $g_{i_1,\ldots,i_k}(s)$ are listed in the lexicographic order of their indexes $(i_1,\ldots,i_k)$. Thus (7) can be expressed as

$$\begin{aligned}
\underline{G}_1(s) &= \mathbf{A}_1(s)\underline{G}_0 + \mathbf{B}_1(s)\underline{G}_2(s),\\
&\vdots\\
\underline{G}_k(s) &= \mathbf{A}_k(s)\underline{G}_{k-1}(s) + \mathbf{B}_k\underline{G}_{k+1}(s),\\
&\vdots\\
\underline{G}_n(s) &= \mathbf{A}_n(s)\underline{G}_{n-1}(s),
\end{aligned} \tag{8}$$

where $\underline{G}_0=(\mu_1, \ldots, \mu_n)^*$, the matrix $\mathbf{A}_k(s)$ is of order $\binom{n}{k}\times\binom{n}{k-1}$, $k=1, \ldots, n$ $\mathbf{B}_k(s)$ is of order $\binom{n}{k}\times\binom{n}{k+1}$, $k=0, 1, \ldots, n-1$. The elements of $\mathbf{A}_k(s)$, $\mathbf{B}_k(s)$ can be readily determined with the aid of eq. (7). It is easy to see that the solution to (8) can be evaluated in the same way as in (5), that is

$$\underline{G}_k(s) = \mathbf{F}_k(s)\underline{G}_{k-1}(s), \tag{9}$$

where

$$\mathbf{F}_n(s)= \mathbf{A}_n(s),\ \mathbf{F}_k(s) = (1-\mathbf{B}_k(s)\mathbf{F}_{k+1}(s))^{-1}\mathbf{A}_k(s), \quad k = 1, \ldots, n-1.$$

Finally,

$$Me^{-s\delta} = \sum \frac{\lambda_i}{\Lambda} g_i(s).$$

The moments of busy period δ can be obtained from eq. (8) by differentiating and setting $s=0$. If $\mathbf{A}^{(i)}(s)$ denotes the i-th derivate of matrix $\mathbf{A}(s)$, which is meant by elements, then it is easy to see that

$$(\mathbf{A}(s)\mathbf{B}(s))^{(i)} = \sum_{l=0}^{i}\binom{i}{l}\mathbf{A}^{(l)}(s)\mathbf{B}^{(i-l)}(s).$$

If we define $M^{(i)}_{i_1,\ldots,i_k}$ and $\underline{M}^{(i)}_k$ by

$$M^{(i)}_{i_1,\ldots,i_k} = (-1)^i \frac{d^i g_{i_1,\ldots,i_k}(s)}{ds^i}\bigg|_{s=0},$$

and

$$\underline{M}^{(i)}_k = (-1)^i \underline{G}^{(i)}_k(0),$$

then

$$\underline{G}^{(i)}_k(0) = \sum_{l=0}^{i}\binom{i}{l}\mathbf{A}^{(l)}_k(0)\underline{G}^{(i-l)}_{k-1}(0) + \sum_{l=0}^{i}\binom{i}{l}\mathbf{B}^{(l)}_k(0)\underline{G}^{(i-l)}_{k+1}(0),$$

which yields

$$\underline{M}^{(i)}_k = \sum_{l=0}^{i}(-1)^l\binom{i}{l}\mathbf{A}^{(l)}_k(0)\underline{M}^{(i-l)}_{k-1} + \sum_{l=0}^{i}(-1)^l\binom{i}{l}\mathbf{B}^{(l)}_k(0)\underline{M}^{(i-l)}_{k+1}. \tag{10}$$

Introducing

$$\mathbf{A}^{(l)}_k = \mathbf{A}^{(0)}_k(0), \quad \mathbf{B}^{(i)}_k = \mathbf{B}^{(0)}_k(0), \quad \underline{C}^{(i)}_k = \sum_{l=1}^{i}(-1)^l\binom{i}{l}[\mathbf{A}^{(l)}_k(0)\underline{M}^{(i-l)}_{k-1} + \mathbf{B}^{(l)}_k(0)M^{(i-l)}_{k+1}]$$

(10) can be written as

$$\underline{M}^{(i)}_k = \mathbf{A}^{(i)}_k\underline{M}^{(i)}_{k-1} + \mathbf{B}^{(i)}_k\underline{M}^{(i)}_{k+1} + \underline{C}^{(i)}_k.$$

Thus, the equations for the i-th moment of the busy period in matrix form are

$$\begin{aligned}
&\underline{M}^{(i)}_1 = \mathbf{B}^{(i)}_1\underline{M}^{(i)}_2 + \underline{C}^{(i)}_1,\\
&\vdots\\
&\underline{M}^{(i)}_k = \mathbf{A}^{(i)}_k\underline{M}^{(i)}_{k-1} + \mathbf{B}^{(i)}_k\underline{M}^{(i)}_{k+1} + \underline{C}^{(i)}_k,\\
&\vdots\\
&\underline{M}^{(i)}_n = \mathbf{A}^{(i)}_n\underline{M}^{(i)}_{n-1} + \underline{C}^{(i)}_n.
\end{aligned} \tag{11}$$

In the following we show that the solution to (11) can be obtained as

$$\underline{M}_k^{(i)} = F_k^{(i)} \underline{M}_{k-1}^{(i)} + \underline{D}_k^{(i)}.$$

To derive this, define $\mathbf{F}_n^{(i)}$ and $\underline{D}_n^{(i)}$ by

$$\mathbf{F}_n^{(i)} = \mathbf{A}_n^{(i)}, \quad \underline{D}_n^{(i)} = \underline{C}_n^{(i)}. \tag{12}$$

Assuming that $M_{k+1}^{(i)} = \mathbf{F}_{k+1}^{(i)} \underline{M}_k^{(i)} + \underline{D}_{k+1}^{(i)}$, after substituting to (11) we get

$$(1 - \mathbf{B}_k^{(i)} \mathbf{F}_{k+1}^{(i)}) \underline{M}_k^{(i)} = \mathbf{A}_k^{(i)} \underline{M}_{k-1}^{(i)} + \mathbf{B}_k^{(i)} \underline{D}_{k+1}^{(i)} + \underline{C}_k^{(i)},$$

which yields

$$\mathbf{F}_k^{(i)} = (1 - \mathbf{B}_k^{(i)} \mathbf{F}_{k+1}^{(i)})^{-1} \mathbf{A}_k^{(i)}, \quad \underline{D}_k^{(i)} = (1 - \mathbf{B}_k^{(i)} \mathbf{F}_{k+1}^{(i)})^{-1} (\mathbf{B}_k^{(i)} \underline{D}_{k+1}^{(i)} + \underline{C}_k^{(i)}). \tag{13}$$

Concluding

$$\begin{aligned} \underline{M}_1^{(i)} &= \underline{D}_1^{(i)}, \\ &\vdots \\ \underline{M}_k^{(i)} &= \mathbf{F}_k^{(i)} \underline{M}_{k-1}^{(i)} + \underline{D}_k^{(i)}, \quad k = 2, \ldots, n, \end{aligned}$$

where matrix $\mathbf{F}_k^{(i)}$ and vector $\underline{D}_k^{(i)}$ are defined by (12), (13). Finally,

$$M\delta^{(i)} = \sum \frac{\lambda_l}{\Lambda} M\delta_l^{(i)}. \tag{14}$$

In particular, if $i=1$, (14) reduces to equations found in Tomkó [10].

(ii) *Response times.* Let $\{\tau_k, k \geqq 0\}$ denote the instants of consecutive changes of states in Markov chain $(\underline{x}(t), t \geqq 0)$. Let us consider the following imbedded Markov chain $(\underline{Y}_m, m \geqq 0)$ defined by $\underline{Y}_m = \underline{X}(\tau_m + 0)$. If we define by

$$(q_0, q_{i_1, \ldots, i_k}; (i_1, \ldots, i_k) \in c_k^n, \ k = 1, \ldots, n) \tag{15}$$

the stationary distribution of $(\underline{Y}_m, m \geqq 0)$ then it is clear that

$$P_0 = \frac{q_0}{\Lambda} \Big/ \left(q_0/\Lambda + \sum_{l=1}^{n} \sum_{c_l^n} \frac{q_{i_1, \ldots, i_l}}{\sigma_{i_1, \ldots, i_l}} \right),$$

$$P_{i_1, \ldots, i_k} = \frac{q_{i_1, \ldots, i_k}}{\sigma_{i_1, \ldots, i_k}} \Big/ \left(q_0/\Lambda + \sum_{l=1}^{n} \sum_{c_l^n} \frac{q_{i_1, \ldots, i_l}}{\sigma_{i_1, \ldots, i_l}} \right), \tag{16}$$

$k=1, \ldots, n$, (cf. Tomkó [9]).
From (16), for (15) we get the following system of linear equations

$$q_0 \left(\frac{P_0 - 1}{\Lambda} \right) + \sum_l \sum_{c_l^n} \frac{P_0}{\sigma_{i_1, \ldots, i_l}} q_{i_1, \ldots, i_l} = 0,$$

$$\frac{P_{i_1, \ldots, i_k}}{\Lambda} q_0 + \sum_l \sum_{c_l^n, (i_1, \ldots, i_l) \neq (i_1, \ldots, i_k)} \frac{P_{i_1, \ldots, i_k}}{\sigma_{i_1, \ldots, i_l}} q_{i_1, \ldots, i_l} + \frac{P_{i_1, \ldots, i_k} - 1}{\sigma_{i_1, \ldots, i_k}} q_{i_1, \ldots, i_k} = 0,$$

which can be solved by standard methods.

Let $E_{i_1, \ldots, i_k}^{(i)}$ denote the event that at the arrival epoch of job i programs with indexes $(i_1, \ldots, i_k)$ are processed by the CPU, $(i_1, \ldots, i_k) \in C_k^n$, $i_1 \neq i, \ldots, i_k \neq i$, $1 \leqq i \leqq n$,

$0 \leqq k \leqq n-1$. In addition, let $\gamma^{(i)}_{i_1,\ldots,i_k}$ denote the response time of job i if event $E^{(i)}_{i_1,\ldots,i_k}$ occurs. Denote by $q^{(i)}_{i_1,\ldots,i_k}$ the steady-state probability of $E^{(i)}_{i_1,\ldots,i_k}$. Then similarly to Csige—Tomkó [4], for $q^{(i)}_{i_1,\ldots,i_k}$, we have

$$q_0^{(i)} = q_0 \frac{\lambda_i}{\Lambda} Q^{(i)}, \quad q^{(i)}_{i_1,\ldots,i_k} = q_{i_1,\ldots,i_k} \frac{\lambda_i}{\sigma_{i_1,\ldots,i_k}} Q^{(i)},$$

where $Q^{(i)}$ can be determined from the norming condition by

$$Q^{(i)} = \frac{1}{\lambda_i}\left(\frac{q_0}{\Lambda} + \sum_{k=1}^{n-1} \sum_{c_k^n} \frac{q_{i_1,\ldots,i_k}}{\sigma_{i_1,\ldots,i_k}}\right)^{-1}.$$

Using the results derived by Tomkó in [9] the following iterative relations can be written

$$\gamma_0^{(i)} \doteq \eta\sigma_i + \sum_{l \neq i} \gamma_l^{(i)} \chi(\eta_{\lambda_l} = \eta\sigma_i),$$

$$\vdots$$

$$\gamma^{(i)}_{i_1,\ldots,i_k} \doteq \eta\sigma_{i_1,\ldots,i_k,i} + \sum_{l \neq i_1,\ldots,i_k} \gamma^{(i)}_{i'_1,\ldots,i'_{k+1}} \chi(\eta_{\lambda_l} = \eta\sigma_{i_1,\ldots,i_k,i}) +$$

$$+ \sum_{j=1}^{k} \gamma^{(i)}_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_k} \chi\left(\eta \frac{\mu_{i_j} r_{i_j}}{R_{i_1,\ldots,i_k,i}} = \eta\sigma_{i_1,\ldots,i_k,i}\right), \tag{17}$$

$$\gamma^{(i)}_{1,\ldots,i-1,i+1,n} \doteq \eta\sigma_{1,\ldots,n} + \sum_{j=1}^{n} \gamma^{(i)}_{1,\ldots,j-1,j+1,\ldots,n} \chi\left(\eta \frac{\mu_j r_j}{R_{1,\ldots,n}} = \eta\sigma_{1,\ldots,n}\right),$$

Introducing the L—S transform

$$v^{(i)}_{i_1,\ldots,i_k}(s) = Me^{-s\gamma^{(i)}_{i_1,\ldots,i_k}},$$

from (17) we get

$$v_0^{(i)}(s) = \frac{1}{s+\sigma_i}\left[\mu_i + \sum_{l \neq i} \lambda_l v_l^{(i)}(s)\right], \tag{18}$$

$$\vdots$$

$$v^{(i)}_{i_1,\ldots,i_k}(s) = \frac{1}{s+\sigma_{i_1,\ldots,i_k,i}}\left[\sum_{j=1}^{k} \frac{\mu_{ij} r_{ij}}{R_{i_1,\ldots,i_k,i}} v^{(i)}_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_k}(s) + \sum_{l \neq i_1,\ldots,i_k} v^{(i)}_{i'_1,\ldots,i'_{k+1}}(s)\right],$$

$$v^{(i)}_{1,\ldots,i-1,i+1,\ldots,n}(s) = \frac{1}{s+\sigma_{1,\ldots,n}} \sum_{j=1}^{n} \frac{u_j r_j}{R_{1,\ldots,n}} v^{(i)}_{1,\ldots,j-1,j+1,\ldots,n}(s).$$

Finally, the L—S transform of the response time for job i can be easily obtained by

$$v^{(i)}(s) = q_0^{(i)} v_0^{(i)}(s) + \sum_{k=1}^{n-1} \sum_{c_k^n} q^{(i)}_{i_1,\ldots,i_k} v^{(i)}_{i_1,\ldots,i_k}(s).$$

Notice that eq. (18) can be treated in the same way as eq. (7).

5. Numerical results

The algorithm generating these performance measures were implemented in PL/1 in the Computer Centre of University of Debrecen. Some sample results for different input parameters λ_i, μ_i, n $(i=1, \ldots, n)$ are shown in Tables 1—4. In Table 1 and 2 some comparisons are made with PS and PR disciplines, while in Table 3 and 4 we give the characteristics under PPS discipline.

Table. 1. Homogeneous I/0 and CPU times

Parameters: $\lambda_1=\lambda_2=\lambda_3=0.3$, $\mu_1=\mu_2=\mu_3=0.7$

	R_i	T_i	$\hat{W}_i$	U	$M\delta$	$\bar{n}$	$\hat{W}$	T
PR	1.428	0.21	0					
	2.413	0.17	0.689	0.74	3.177	1.27	2.658	0.51
	4.242	0.13	1.969					
PS	2.450	0.17	0.715					
	2.450	0.17	0.715	0.74	3.177	1.27	2.145	0.51
	2.450	0.17	0.715					
PPS weights								
125	1.467	0.21	0.026					
5	2.559	0.16	0.791	0.74	3.177	1.27	2.460	0.51
1	3.776	0.14	1.643					
1000	1.437	0.21	0.005					
10	2.485	0.17	0.739	0.74	3.177	1.27	2.512	0.51
1	3.955	0.13	1.768					
125 000	1.428	0.21	0.000					
50	2.396	0.17	0.677	0.74	3.177	1.27	2.561	0.51
1	4.120	0.13	1.884					
1 000 000	1.428	0.21	0.000					
100	2.383	0.17	0.668	0.74	3.177	1.27	2.568	0.51
1	4.143	0.13	1.900					

Table 2. Heterogeneous I/0 and CPU times

Parameters: $\lambda_1=0.5$, $\lambda_2=0.3$, $\lambda_3=0.2$
$\mu_1=0.9$, $\mu_2=0.7$, $\mu_3=0.5$

PR	1.125	0.32	0					
	2.619	0.16	0.833	0.77	3.34	1.34	3.014	0.56
	6.363	0.08	2.181					
PS	1.885	0.25	0.697					
	2.526	0.17	0.768	0.76	3.21	1.33	2.252	0.53
	3.574	0.11	0.787					
PPS								
1	2.056	0.24	0.850					
1	2.745	0.16	0.921	0.75	3.15	1.33	2.266	0.52
2	2.990	0.12	0.495					

Table 3. Homogeneous I/0 times

Parameters: $\lambda_1=\lambda_2=\lambda_3=0.2$, $\mu_1=0.4$, $\mu_2=0.6$, $\mu_3=0.8$

weights								
1	4.831	0.10	0.932					
5	2.498	0.13	0.499	0.675	3.472	1.028	1.453	0.394
125	1.277	0.15	0.021					
1	4.965	0.10	0.986					
10	2.407	0.13	0.444	0.675	3.472	1.024	1.435	0.395
100	1.256	0.15	0.004					
1	5.100	0.09	1.040					
100	2.304	0.13	0.382	0.675	3.472	1.020	1.423	0.395
1 000 000	1.250	0.15	0.000					

Table 4. Heterogeneous I/0 and CPU times

Parameters: $\lambda_1=1$, $\lambda_2=2$, $\lambda_3=3$, $\lambda_4=4$, $\lambda_5=5$, $\lambda_6=6$
$\mu_1=6$, $\mu_2=5$, $\mu_3=4$, $\mu_4=3$, $\mu_5=2$, $\mu_6=1$

weights								
1	3.008	0.24	17.048					
2	1.834	0.42	8.170					
3	1.543	0.53	5.172	0.999	289.350	5.070	38.795	2.525
4	1.549	0.55	3.647					
5	1.853	0.48	2.706					
6	3.052	0.31	2.052					
36	0.329	0.75	0.974					
25	0.435	1.06	1.175					
16	0.659	1.00	1.636	0.999	94.777	4.145	33.764	3.775
9	1.233	0.65	2.699					
4	3.379	0.27	5.758					
1	22.522	0.04	21.522					

Conclusion

In this paper we have modelled a multiprogrammed computer system as finite-source single server queueing system with different types of customers under priority processor-shared service discipline. The system performance measures were numerically evaluated using an algorithmic approach.

Acknowledgement. The numerical results were obtained by A. Pósafalvi whom I am very grateful. My thanks are also due to Prof. M. Arató for several helpful comments.

Abstract

This paper deals with a heterogeneous multiprogrammed computer system under priority processor-shared (PPS) service discipline introduced by Kleinrock. The jobs are characterized by exponentially distributed input-output (I/0) and central processing unit (CPU) times. In steady state the main performance measures, such as CPU utilization, expected CPU busy period length, mean response times, waiting ratio, throughput of the jobs and throughput of the system, are given.

In addition, a system of linear equations for Laplace—Stieltjes (L—S) transform of the response times and the CPU busy period length is obtained. Finally, by numerical examples characteristics under different service disciplines are compared with the PPS one.

Keywords: I/0 times, CPU times, utilization, mean response time, waiting ratio, throughput.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF DEBRECEN
PF. 12. DEBRECEN, HUNGARY
H-4010

References

[1] D. Asztalos, Finite-Source Queueing Models and their Applications to Computer Systems, Alk. Mat. Lapok 5 (1978), 89—101 (in Hungarian).
[2] B. Avi-Itzhak and D. P. Heyman, Approximate Queueing Models for Multiprogramming Computer Systems, Opns. Res. 21 (1973), 1212—1230.
[3] J. W. Cohen, The Multiple Phase Service Network with Generalized Processor Sharing, Acta Informatica 12 (1979), 245—284.
[4] L. Csige and J. Tomkó, The machine interference for exponentially distributed operating and repair times, Alk. Mat. Lapok 8 (1982), 107—124 (in Hungarian).
[5] L. Kleinrock, Queueing Systems, Vol. 2: Computer Applications, Wiley-Interscience, New York, 1976.
[6] L. Kleinrock, Time-Shared Systems: A Theoretical Treatment, J.ACM 14 (1967), 242—261.
[7] J. P. Lehoczky and D. P. Gaver, Models for Time-Sharing Computer Systems with Heterogeneous Users, Opns. Res. 29 (1981), 550—566.
[8] J. Sztrik, Probability Model for non-homogeneous Multiprogramming Computer System, Acta Cybernetica 6 (1983), 93—101.
[9] J. Tomkó, Sojourn time problems for Markov chains, Alk. Mat. Lapok 8 (1982), 91—106 (in Hungarian).
[10] J. Tomkó, CPU utilization study II, Alk. Mat. Lapok 3 (1977), 83—96 (in Hungarian).

(Received March 13, 1985)

85-2217 — Szegedi Nyomda — F. v.: Dobó József igazgató

Iterative Method for Solving M/G/1//N-type Loops with Priority Queues

By ANDRÁS SZÉP

1. Introduction

The range of applicability of queueing models increases every year. Here, closed queueing systems with general service time distributions and several priority dispatching rules are of special interest. However, existing solution methods such as [1] for a special case of exponential servers are cumbersome and cannot be applicable to cases of general distributions. On other hand, an original algorithm [2] to solve $M/G/1//N$-type loops with *FCFS* (first-come-first-served) queues has been recently suggested. Through combining this solution technique with an effective method of decompositions of general servers into exponentials [3] a really well-work method can be synthetized. It would be of practical significant if this technique could be applicable to cases of priority dispatching rules at queues.

In this paper an iterational method is suggested for performance evaluation of closed queueing systems with preemptive resume and non-preemptive priority queues and general service time distributions.

2. Preemptive resume priority queues

Consider a closed queueing system consisting of two service centers, at one of which there is exponential service time distribution and an infinite number of servers (i.e. simple delay type service center), and the other is a Coxian collection replacement for an arbitrary non-exponential server with preemptive resume queueing discipline (see fig. 1). Assume that customers belong to one of R priority classes and

(i) class p customers have priority over class r customers at phase I if $1 \leqq p < r \leqq R$,

(ii) customers whithin the the same priority class follow the *FCFS* queueing discipline at phase I.

At second phase all customers are served simultaneously due to an infinite number of servers but service rates for different classes' customers may vary.

For convenience, we define

R — number of customers classes,

n_p — number of customers in class p $(1 \leqq p \leqq R)$,

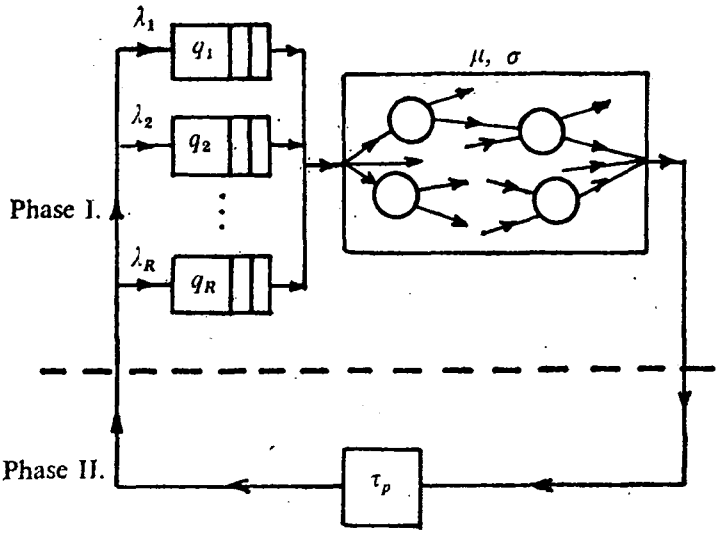


Fig. 1
An $M/G/1//N$-type loop with priority queueing discipline

n_p^* — total number of customers in first p classes,

$$n_p^* = \sum_{r=1}^{p} n_r, \quad \text{for} \quad p = 1, 2, \ldots R, \tag{1}$$

μ — mean service rate at phase I,
σ — standard deviation of service time at phase I,
τ_p — mean service time of customers of class p at phase II, $(1 \leqq p \leqq R)$,
τ_p^* — aggregate mean service time of customers of first p classes at phase II , $(1 \leqq p \leqq R)$,
λ_p — throughput of customers of class p, $(1 \leqq p \leqq R)$,
λ_p^* — aggregate throughput of customers of first p classes

$$\lambda_p^* = \sum_{r=1}^{p} \lambda_r, \quad \text{for} \quad p = 1, 2, \ldots R, \tag{2}$$

q_p — average number of customers of class p at phase I,
q_p^* — aggregate average number of customers of first p classes at phase I,

$$q_p^* = \sum_{r=1}^{p} q_r, \quad \text{for} \quad p = 1, 2, \ldots R, \tag{3}$$

t_p — average elapsed time of customers at phase I for class p, $(1 \leqq p \leqq R)$,
t_p^* — aggregate mean elapsed time of customers of first p classes at phase I, $(1 \leqq p \leqq R)$.

The method suggested in this paper is based on the approach described in [4]. Thus, Little's rule [5] may be applied to the queueing system being considered

$$q_p = \lambda_p t_p, \quad \text{for} \quad p = 1, 2, \ldots R. \tag{4}$$

Several authors (e.g. Jaiswal [6]) noted that aggregate performance characteristics such as mean response time etc. do not depend on the queueing discipline. Moreover, in case of preemptive resume priority disciplines performance characteristics of servicing low priority customers (r) have no effect on servicing high priority customers (p) ($1 \leqq p < r \leqq R$). Therefore, Little's rule may be applied to the aggregate characteristics

$$q_p^* = \lambda_p^* t_p^*, \quad \text{for} \quad p = 1, 2, \ldots R. \tag{5}$$

Having made some simple transformations on (1)—(5) we obtain

$$t_p = \frac{\lambda_p^* t_p^* - \lambda_{p-1}^* t_{p-1}^*}{\lambda_p^* - \lambda_{p-1}^*}, \qquad \text{for} \quad p = 1, 2, \ldots R, \tag{6}$$

assuming $\lambda_0^* = 0$ and $t_0^* = 0$.

The last expression shows the way of successive computation of serving characteristics for customers at all levels of priorities. To attain this it is enough to compute aggregate serving characteristics for the same queueing system as given but with *FCFS* serving discipline at phase I.

There are well known methods and formulas for performance evaluation of closed queueing system with exponential servers. Recently an effective algorithm has been suggested to solve $M/G/1//N$-type loops with *FCFS* queueing discipline in [2].

The above mentioned problem can be solved though difficulties arise. The quantities λ_p and t_p^* are given by Little's formula (5):

$$\lambda_p = \frac{n_p}{\tau_p + t_p}, \quad \text{for} \quad p = 1, 2, \ldots R, \tag{7}$$

and

$$\tau_p^* = \frac{n_p^*}{\lambda_p^*} - t_p^*, \quad \text{for} \quad p = 1, 2, \ldots R. \tag{8}$$

Since t_p does not depend on λ_p linearly, an iterational method can be suggested. The following algorithm contains two embedded iteration processes.

Algorithm 1.

Step 1. Input data — R,

$$\text{and} \quad \|n_1, n_2, \ldots n_R\|, \quad \|\tau_1, \tau_2, \ldots \tau_R\|, \quad \mu, \sigma.$$

Step 2. Initial values $p \leftarrow 0$,

$$\text{and} \quad n_0^* \leftarrow 0, \quad \lambda_0^* \leftarrow 0, \quad t_0^* \leftarrow 0.$$

Step 3. Get next class $p \leftarrow p+1$,

$$\text{take} \quad n_p^* \leftarrow n_{p-1}^* + n_p,$$

$$\text{and first approximation} \quad t_p \leftarrow \frac{n_p}{\mu}.$$

Step 4. Compute

$$\lambda_p \leftarrow \frac{n_p}{\tau_p + t_p},$$

$$\text{and} \quad \lambda_p^* \leftarrow \lambda_{p-1}^* + \lambda_p,$$

$$\text{and first approximation} \quad t_p^* \leftarrow \frac{n_p^*}{\mu}.$$

Step 5. Find next approximation $\tau_p^* \leftarrow \frac{n_p^*}{\lambda_p^*} - t_p^*$.

Step 6. Knowing n_p^*, τ_p^*, μ and σ find a solution of $M/G/1//N$-type loop with *FCFS* queues by the method suggested in [2] and obtain new value for $t_p^{*\prime}$.

Step 7. If $|t_p^{*\prime} - t_p^*| > \varepsilon$ then take $t_p^* \leftarrow t_p^{*\prime}$ and go to step 5., else compute t_p' by formula (6).

Step 8. If $|t_p' - t_p| > \varepsilon$ then take $t_p \leftarrow t_p'$ and go to step 4., else compute $q_p \leftarrow \lambda_p \cdot t_p$.

Step 9. If $p < R$ then go to step 3., else output results —

$$\|\lambda_1, \lambda_2, \ldots \lambda_R\|, \quad \|t_1, t_2, \ldots t_R\|, \quad \|q_1, q_2, \ldots q_R\| \text{ and stop.}$$

(ε — means the error's bound)

It is easy to prove that both iterational processes of this algorithm converge (see fig. 2. and Appendix *A*). The number of elementary operations required for computation is equal to $\sigma(i_1, i_2, N)$, where

$$N = \sum_{r=1}^{R} \sum_{p=1}^{r} n_p,$$

and i_1, i_2-means the number of iterations.

For $\varepsilon = 0.1\%$ the total number of iterations (i_1, i_2) in most cases did not exceed 30, therefore the suggested method for the performance evaluation of closed queueing systems with priorities looks much more efficient than the methods based on calculations of all steady-state probabilities of the system. Note that the number of states of such systems is around $\sim N^R$.

3. Computation speed

A further study of iterational processes in Algorithm 1 indicates that although in most cases they converge rapidly, in case of heavy traffic a relatively large number of iterations may be required (> 100). Therefore we expect it to increase the speed of convergence. This can be achieved by several ways. First, if one chooses a better first approximation, secondly by using more powerful solution searching methods (dichotomic, gradient etc.), and at last by merging two iteration processes. Experiments show that the simultaneous implementation of first and third principle provides the maximum increase of computation performance. Implementation of the second way causes an additional consumption in use of computer resources.

The next algorithm for better convergence was derived for getting performance

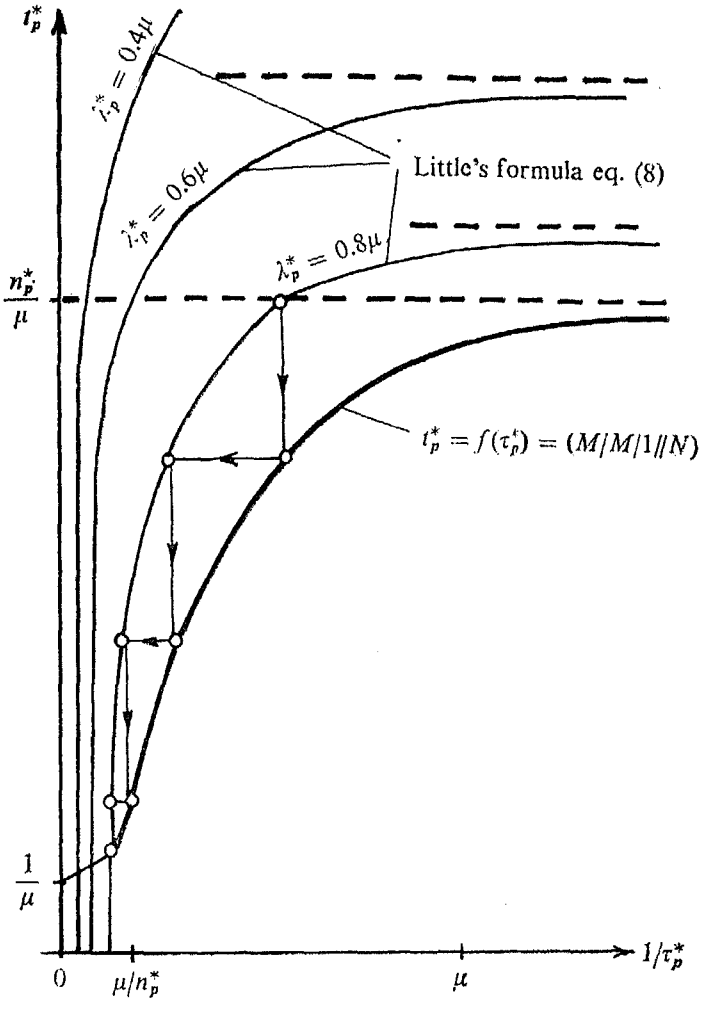


Fig. 2
Dependence of the elapsed times of customers at phase I upon the serving rate at phase II, and an example of convergence

of closed queueing loops with generalized service time distribution and preemptive resume priority dispatching rules.

Algorithm 2.

Step 1. Input data – R,

$$\text{and} \quad \|n_1, n_2, \ldots n_R\|; \ \|\tau_1, \tau_2, \ldots \tau_R\|, \quad \mu, \sigma.$$

Step 2. Initial values $p \leftarrow 0$,

$$\text{and} \quad n_0^* \leftarrow 0, \quad \lambda_0^* \leftarrow 0, \quad t_0^* \leftarrow 0.$$

Step 3. Get next class $p \leftarrow p+1$,

$$\text{and take} \quad n_p^* \leftarrow n_{p-1}^* + n_p.$$

Step 4. Find first approximation for t_p knowing n_p, τ_p, μ and σ by the method of [2], and let $t_p^* \leftarrow t_p$.

Step 5. Compute

$$\lambda_p \leftarrow \frac{n_p}{\tau_p + t_p},$$

$$\lambda_p^* \leftarrow \lambda_{p-1}^* + \lambda_p;$$

$$\text{and} \quad \tau_p^* \leftarrow \frac{n_p^*}{\lambda_p^*} - t_p^*.$$

Step 6. Knowing n_p^*, τ_p^*, μ and σ compute t_p^* by the method of [2].

Step 7. Compute t_p' by the formula (6).

Step 8. If $|t_p' - t_p| > \varepsilon$ then let $t_p \leftarrow t_p'$ and go to step 5., else compute $q_p \leftarrow \lambda_p t_p$.

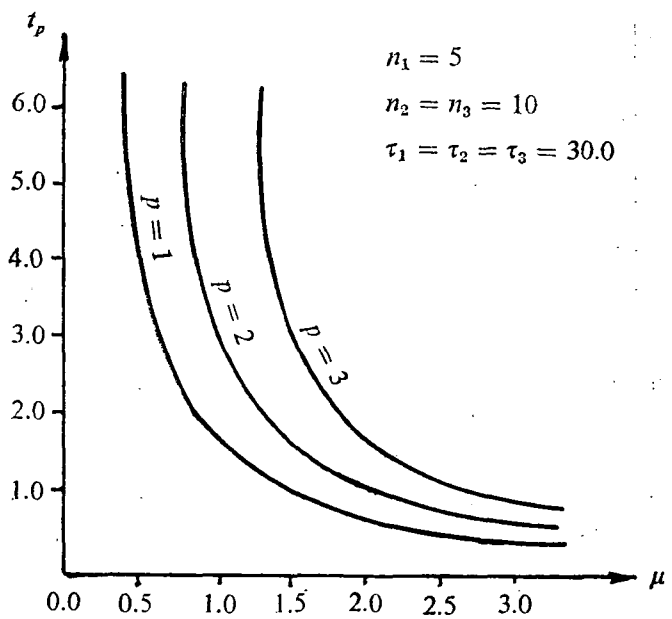


Fig. 3
Dependence of the mean elapsed times of customers upon the service rates at phase I for an $M/G/1//N$-type preemptive resume system

Step 9. If $p<R$ then go to step 3., else output results —

$$\|\lambda_1, \lambda_2, \ldots \lambda_R\|, \quad \|t_1, t_2, \ldots t_R\|, \quad \|q_1, q_2, \ldots q_R\|$$

and stop.

The study of convergence of the above algorithm indicates that 1—10 iterations are sufficient for $\varepsilon=0.1\%$ even in the most extreme cases. Note that for the case of exponential distribution of service times at phase I the mean elapsed times t_p^* can be found by mean-value analysis (MVA) methods (see [7]) at step 6., of Algorithm 1., and at steps 4. and 6., of Algorithm 2.

4. Non-preemptive priority queues

A common approach for solving closed queueing systems with non-preemptive priority queueing discipline is similar to that applied to solve systems with preemptive resume priorities. But it differs since in non-preemptive priority queues the service of customers cannot be interrupted by the arrival of customers with higher priority and they must await releasing of the server. Let us define the mean residual life-time W_p as the mean time which remains until the end of actual service of customer in class p. Then [6]

$$W_p = \frac{\lambda_p \overline{X^2}}{2}, \quad \text{for} \quad p = 1, 2, \ldots R, \tag{9}$$

where $\overline{X^2}$ — means the second moment of the service time distribution. It is clear that the aggregate mean elapsed time of customers in the first p classes at phase I is equal to the aggregate service time of customers in the first p classes in the system with preemptive resume priorities plus the mean residual life-time of all priority classes with number greater than p, i.e.,

$$t_p^*(\text{non-preemptive}) = t_p^*(\text{preemptive}) + \frac{(\lambda_R^* - \lambda_p^*)\overline{X^2}}{2}, \text{ for } p = 1,2,\ldots R \tag{10}$$

where $(\lambda_R^* - \lambda_p^*)$ means the aggregate customers throughput of classes $p+1, p+2, \ldots, R$. Note that λ_R^* (non-preemptive) $= \lambda_R^*$ (preemptive), and the aggregate throughput λ_R^* does not depend on queueing discipline and can be calculated although neither troughputs nor residual lifetimes are known.

The next algorithm is suggested to calculate the performance of closed queueing loops with generalized service time distribution and non-preemptive priority queueing discipline on the basis of Algorithm 2.

Algorithm 3.

Step 1. Do all the claculations of Algorithm 2, and define λ_R^*.

Step 2. Do once more all the calculations of Algorithm 2 with the exception of a correction at step 6, where to the computed value of t_p^* the residual life-times are added

$$t_p^* \leftarrow t_p^* + \frac{(\lambda_R^* - \lambda_p^*)\overline{X^2}}{2}, \quad (p = 1, 2, \ldots R).$$

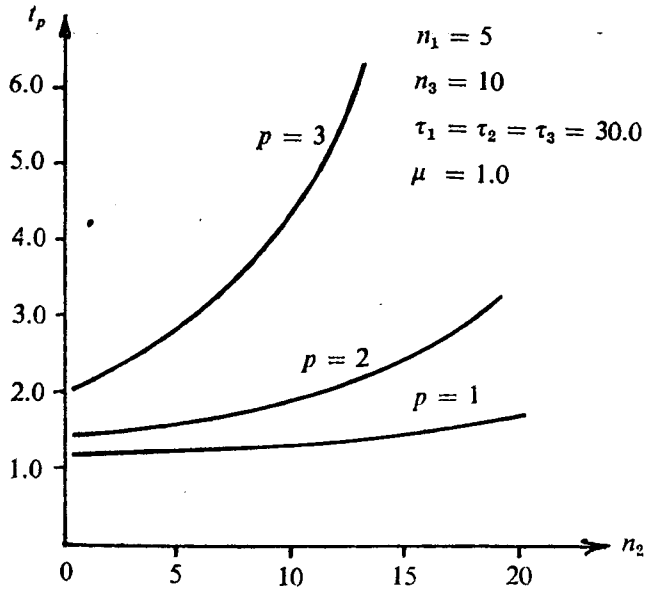


Fig. 4
Dependence of the mean elapsed times upon the number of customers in the second class

This algorithm has the same advantages as the previous one.

In conclusion note that if service rates at phase I for all classes are equal then the suggested method and described algorithms will ensure obtaining theoretically accurate results. If service rates are different for different classes of customers then the aggregate service rates have to be evaluated by

$$\mu_p = \frac{\sum_{r=1}^{p} \lambda_r \mu_r}{\lambda_p^*}, \qquad \text{for} \quad p = 1, 2, \ldots R. \tag{11}$$

Although this way allows for systematic errors in the results of computations, usually evaluated quantities of modelled systems remain within the range of applicability and errors do not exceed 10% [4].

5. Conclusion

Simple algorithms for computing exact mean elapsed times, queue lengths and throughputs of individual customers classes in $M/G/1//N$-type closed preemptive resume and non-preemptive priority systems have been presented for the case when all customers classes have equal mean service times at the non-exponential phase and different at the other phase. It has been shown that algorithms based on the

suggested iterational method converge rapidly and they have unique solutions. For the case of unequal service times an approximation technique has been suggested. The described method and algorithms are efficient for solving large scale application problems.

Acknowledgement

I would like to express my deep gratitude to prof. V. I. Dmitriev for his consulting and valuable advices.

Appendix A

Proof of the convergence of iterational processes and the uniqueness of solution.

On fig. 2, the dependence of elapsed times at phase I upon serving times at phase II is shown. Because the linearity in Little's formula for the convergence of our iterations and uniqueness of the solution it is sufficient to prove that $\frac{\partial t_p^*}{\partial \tau_p^*} > -1$.

The proof will be carried out by induction.

According to MVA (see [7])

$$t_p^*(i) = \frac{1}{\mu}\left(1+q_p^*(i-1)\right) = \frac{1}{\mu}\left(1+\lambda_p^*(i-1)t_p^*(i-1)\right) =$$

$$= \frac{1}{\mu}\left[1+\frac{i-1}{\tau_p^*+t_p^*(i-1)}\,t_p^*(i-1\right] = \frac{1}{\mu}\left[1+\frac{i-1}{\dfrac{\tau_p^*}{1+t_p^*(i-1)}}\right].$$

It is obvious that $\frac{\partial t_p^*(0)}{\partial \tau_p^*} = 0 > -1$.

Let us suppose that $\frac{\partial t_p^*(i-1)}{\partial \tau_p^*} > -1$ then

$$\frac{\partial t_p^*(i)}{\partial \tau_p^*} = -\frac{i-1}{\mu}\left[\frac{t_p^*(i-1)-\tau_p^*\dfrac{\partial t_p^*(i-1)}{\partial \tau_p^*}}{(t_p^*(i-1)+\tau_p^*)^2}\right] >$$

$$> -\frac{i-1}{\mu}\,\frac{t_p^*(i-1)+\tau_p^*}{(t_p^*(i-1)+\tau_p^*)^2} = -\frac{i-1}{\mu(t_p^*(i-1)+\tau_p^*)};$$

because $\tau_p^* \cdot \frac{\partial t_p^*(i-1)}{\partial \tau_p^*} > -\tau_p^*$. But $t_p^*(i-1)+\tau_p^* = \frac{i-1}{\lambda_p^*(i-1)}$ and $\lambda_p^*(i-1) \leqq \mu$ which gives

$$\frac{\partial t_p^*(i)}{\partial \tau_p^*} > -1. \quad \square$$

Abstract

Based on Little's formula an iterational method was derived for the solution of $M/G/1//N$-type closed queueing systems with preemptive resume and non-preemptive priority queues at the general server. Efficient algorithms are outlined and described in detail. Convergence of iterations and uniqueness of solution was proved and also an approximation technique was suggested for the case when service rates differ for different classes of customers at the general server.

Keywords. Closed queueing systems, general distributions, preemptive resume and non-preemptive priorities, iterational method, mean-value analysis.

CENTRAL RESEARCH INSTITUTE FOR PHYSICS
OF THE HUNGARIAN ACADEMY OF SCIENCES
P.O.BOX 49, BUDAPEST, H-1525, HUNGARY

References

[1] A. BRANDWAJN, A finite difference equations approach to a priority queue, Opns. Res. 30. (1982), 74—81.
[2] J. L. CAROLL, A. VAN DE LIEFVOORT, L. LIPSKY, Solutions of $M/G/1//N$-type loops with extensions to $M/G/1$ and $GI/M/1$ queue, Opns. Res. 30 (1982), 490—514.
[3] K. C. SEVCIK, A. I. LEVY, S. K. TRIPATHI, J. L. ZAHORJAN, Improving approximations of aggregated queueing network subsystems, in: Computer Performance (Proc. Int. Symp. on Computer Performance Modeling and Evaluation, New York, 1977, North-Holland publ. co. Amsterdam, 1977), 1—22.
[4] J. P. BUZEN, A. B. BONDI, The response times of priority classes under preemptive resume in $M/M/m$ queues, Opns. Res. 31 (1983), 456—465.
[5] J. D. C. LITTLE, A proof for the queueing formula $L=\lambda W$, Opns. Res. 9 (1961), 383—387.
[6] N. K. JAISWAL, Priority Queues (Academic Press, New York, 1968).
[7] M. REISER, S. S. LAVENBERG, Mean-value analysis of closed multichain queueing networks, J. ACM 27 (1980) 313—322.

Received Nov. 14, 1984.

Bibliographie

P. Massie: Programming IBM Assembly Language & Instructor's Manual To Accompany "Programming IBM Assembly Language", XIII+342 & I+149 pages, Burgess Communications 1985.

The first book is a comprehensive introduction to IBM assembler language programming. Unlike many books on the subject, it is not a reference manual. "It is designed to carefully balance concepts with practical applications at a level that students will find most useful. Normally students will have completed at least one course in programming before the assembler language course, and will be familiar with many of the basic concepts involved. With this book, experience is not absolutely necessary, since programming steps are presented fully in a logical, building-block sequence. To help build confidence and mastery of the central concepts, students are encouraged to run programs early".

The first book is divided into six parts and twenty-four chapters.

PART ONE, "INTRODUCTION", covers the basic concepts necessary for assembler language programming, but different from high-level languages. Operating system interfaces can be a complex topic, and the initial presentation in this section is done in a simple fashion. This allows the student to begin writing programs quickly, without a full understanding of these interfaces. Dumps are initially used to obtain output from the programs, so the student becomes familiar with reading and understanding dumps at an early stage.

PART TWO, "REGISTER-BASED INSTRUCTIONS", presents register instructions in a simple format designed for easy comprehension by the beginner student. In addition, this section includes a chapter on the simpler methods of developing loops in assembler language. Although structured programming involves much more than simply reducing branch statements in programs, it is important to introduce the basic concepts of structured programming early.

PART THREE, "CHARACTER PROCESSING", covers the different data formats available, the normal I/O macros, and simple character-processing instructions. With the building-blocks of the earlier parts of the book, students should be ready to handle these more complex subjects. The I/O chapter does not present I/O macros exhaustively, but at a level such that students can run programs.

PART FOUR, "PACKED DECIMAL", covers the packed decimal instructions. This is a major family of arithmetic-processing instructions different in concept from the fixed-point binary instructions covered earlier.

PART FIVE, "GENERAL TOPICS", is a carefully-selected collection of chapters on distinct subjects, which can be covered in connection with other chapters, or completely omitted. Many of these topics are advanced, and can be assigned in accordance with the level and interest of the student.

PART SIX, "ADVANCED TOPICS", covers macros and conditional assembly. The intention is to provide an overview of these advanced features of the IBM assembler.

The APPENDICES also contain a sample JCL, the complete instruction set, and a glossary of selected terms necessary for assembler language programming.

To help students master concepts and to reinforce the logical bulding-block structure of the book, each chapter contains a series of important learning aids.

- KEY CONCEPTS opens each chapter and focus student attention on what is to be learned.

— INTRODUCTIONS provide an overview of each chapter and summarize their central concepts.
— KEY TERMS are presented in boldface when introduced, and full definitions are provided in the glossary.
— EXAMPLES are presented in colored boxes to ensure student engagement.
— EXERCISES are introduced thoughout each chapter to help students assess their mastery of the material.
— PROGRAMS, both partial and complete sample listings, are set off by horizontal rules to focus student attention.
— CHAPTER SUMMARIES review terms and concepts, and provide students with study devices in preparation for classroom discussions and examinations.
— the GLOSSARY of terms of the end of the book fully defines terms, and provides an additional study aid for mastery of the material.

The second book, the Instructor's Manual, offers complete support for the instructor's classroom presentations. The manual provides the instructor with the following aids:

— Complete chapter summaries
— Chapter objectives
— Answers to chapter exercises
— Multiple choice and true/false examination questions.

The Instructor's Manual provides as much assistance as possible to the instructor of the course.

These very well-written books can be recommended to students, instructors and everyone interested in programming IBM assembly language.

K. Dévényi (Szeged)

D. F. Stubbs and N. W. Webre, Data Structures with Abstract Data Types and Pascal, ,XVIII+ +459 pages, Brooks/Cole Publishing Company, Monterey, California, 1985.

"This text represents a fresh approach to a first course in data structures. During the past several years, researchers have developed a new method of designing data structures, the new forms of wich are called abstract data types. In the original research papers, abstract data types are quite formal and unsuited to the beginning student. When the formality is removed, however, there remains a basic and easily understood essence.

Our major challenge in writing this book was to make the notion abstract data types an integral part of the study of data structures. At the end of the course, we want students to have knowledge of 'classical' data structures, as well as skills in the use of abstraction, specification, and program construction using modules, or packages. These skills prepare students to make use of the data abstraction facilities of languages such as Ada, Modula 2, and Mesa. They are also an excellent complement to the techniques of top-down design and structured programming. This text prepares students for advanced courses in which the methods are more formal. In addition, it will serve as an important first step toward object-oriented programming."

I certainly recommend the book as a possible text for a first course in data structures.

Gy. Horváth (Szeged)

S. Grier, Pascal for the 80s, XVI+540 pages, Brooks/Cole Publishing Company, Monterey, California, 1985.

This book is designed to serve as an introductory text in programming for college students. It does not attempt to teach a complete knowledge of Pascal, nor to make any student a mature or expert programmer. Advanced topics in Pascal as sets, pointers and recursion are briefly discussed. This text provides an understandig of very simple algorithms. The main emphasis is on the concept of teaching problem-solving in parallel with the programming language.

The book is recommended for people who are interested in teaching programming at a non-academic level.

Gy. Horváth (Szeged)

Natural language parsing. Psychological, computational, and theoretical perspectives (Edited by D. R. Dowty, L. Karttunen, A. M. Zwicky) XIII+413 pages, Cambridge University Press, 1985.

The volume is a collection of papers written by leading researchers in experimental psychology, theoretical linguistics, and artificial inteligence. The contributions are: Introduction (Lauri Karttunen and Arnold M. Zwicky), Measuring syntactic complexity relative to discourse context (Alice Davison and Richard Lutz), Interpreting questions (Elisabet Engdahl), How can grammars help parsers? (Stephen Crain and Janet Dean Fodor,) Syntactic complexity (Lyn Frazier), Processing of sentences with intrasentential code switching (Aravind K. Joshi), Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? (Aravind K. Joshi), Parsing in functional unification grammar (Martin Kay), Parsing in a free word order language (Lauri Karttunen and Martin Kay), A new characterization of attachment preferences (Fernando C. N. Pereira), On not being led up the garden path: the use of context by the psychological syntax processor (Stephen Crain and Mark Steedman), and Do listeners compute linguistic representations? (Michael K. Tanenhaus, Greg N. Carlson, and Mark S. Seidenberg).

The book is warmly recommended to linguists, psycholinguists and computer scientists.

F. Gécseg (Szeged)

Victor J. Law: Standard Pascal; an Introduction to Structured Software Design, XVII+558 pages, Wm. C. Brown Publishers, Dubuque Iowa, 1985.

"It is well recognized that an introductory course in computing should do more than teach the syntax of a high-level language. The course described as CS1, Introduction to Programming Methodology, in CACM, October 1984, lists the important topics that should be covered in early computing courses. This text melds these non-language issues and the Pascal language into a cohesive presentation of modern programming methodology for beginning students."

This text is designed to satisfy two major objectives. One goal is to introduce modern principles of programming, which can be used for small and large programming projects. The other objective is to introduce the Pascal language as a vehicle for implementing algorithms designed with the software life cycle approach.

Chapters are segmented into three parts. The first part introduces new concepts. A sample problem is then posed, a specification is written, and an algorithm design is presented. A relatively new algorithm design tool, the structured chart, is used for all of the designs. The second part covers any new Pascal features required to implement the new concepts as introduced in the first part. The third part contains complete programming examples where the new topics are featured. Each example includes requirements, specification, algorithm, design, Pascal code, and testing. Each chapter ends with review questions, self-test exercises, and programming problems. Advanced problems are excluded. The appendixes include, among others, answers to selected self-tests, a summary of Pascal syntax using syntax diagrams, and a comparision of common Pascal dialects.

This book is recommended as a text for an introductory course in computing.

Gy. Horváth (Szeged)

N. Wirth: Programmieren in Modula-2, XIV+220 Seiten, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

Das Buch ist die Übersetzung der englischen Original-Ausgabe: Programming in Modula 2, Third corrected Edition, Springer-Verlag, 1985.

Als Handbuch für die Programmierung in Modula-2 überdeckt der Text praktisch alle Eigenschaften dieser Sprache. Teil 1 umfaßt die elementaren Begriffe Variable, Ausdruck, Zuweisung, bedingte und Wiederholungs-Anweisung sowie die Datenstruktur des Arrays. Teil 2 führt in das wichtige Konzept der Prozeduren bzw. Unterprogramme ein. Beide, Teil 1 und Teil 2 umfassen im Wesentlichen den Stoff eines Einführungskurses in die Programmierung. Teil 3 befaßt sich mit Datentypen und Strukturen. Dies entspricht im Kern dem Inhalt eines weiterführenden Programmierkurses. Teil 4 führt den Begriff des Moduls ein, eines fundamentalen Konzeptes sowohl für den Entwurf großer Programmsysteme als auch für das Arbeiten in einem Team. Als Beispiel für Module werden einige häufig verwendete Hilfsprogramme für Ein- und Ausgabe dargestellt. Teil 5 schließlich beschreibt Möglichkeiten der Systemprogrammierung, Gerätebehandlung und der Multiprogrammierung. Weiterhin werden praktische Hinweise gegeben, wie und wann die einzelnem Hilfsmittel einzusetzen

sind. Dies ist als Richtschnur für den Erwerb eines anständigen Stils der Programmierung und der Systemstrukturierung zu verstehen.

Empfohlen werden kann das Buch all denen, die an Programmierung in irgendeiner Weise interessiert sind.

J. Csirik (Szeged)

H. Bunke: Modellgesteuerte Bildanalyse, VIII + 301 Seiten (Reihe: Leitfäden der angewandten Informatik), B. G. Teubner, Stuttgart 1985.

Das Buch befasst sich mit der automatischen Analyse von Bildern und Bildfolgen mittels eines Digitalrechners. Es gliedert sich in zwei Teile: im ersten Teil werden die Grundlagen der wissensbasierten Bildanalyse angegeben, und der zweite Teil beschäftigt sich mit einem speziellen wissensbasierten Bildanalysesystem, das an der Universität Erlangen-Nürnberg entwickelt wurde.

Bei der wissensbasierten Bildanalyse (im ersten Teil des Buches) werden drei größere Gebiete angesprochen:

— Methoden zur Extraktion elementarer Bestandteile,
— Wissensdarstellung,
— Wissensnutzung.

Das Material jedes einzelnen Gebietes wird durch einführende Beispiele und durch einen sehr breiten und ausführlichen Literaturüberblick ergänzt. Dies und der logische Aufbau des Textes machen diesen Teil des Buches zu einer beispielhaften Einführung in die wissensbasierte Bildanalyse.

Im zweiten Teil der Arbeit wird ein wissensbasiertes System zur Analyse nuklearmedizinisch gewonnener Bildfolgen des menschlichen Herzens detailliert vorgestellt. Das Hauptziel dabei ist: die automatische Ableitung einer Diagnose aus einer Bildfolge. Das System ist modular aus den folgenden Komponenten aufgebaut: Modell, Instanzen, Methoden, Kontrolle und Dialog. Dabei werden die grundlegenden Ideen ausführlich beschreiben und das Konzept der Arbeit übersichtlich vorgestellt. Auch die ersten Ergebnisse des klinischen Einsatzes sind gegeben.

Das Buch passt sehr gut zu den Zielen der Reihe "Leitfäden der angewandten Informatik". Ich empfehle es sowohl für Fachleute eines anderen Gebietes als auch für Informatiker, die auf diesem Gebiet tätig werden wollen.

J. Csirik (Szeged)

Varrel C. Grout: Programming with BASIC, XIII + 362 pages, Wm. C. Brown Publishers, Dubuque, Iowa, 1985.

This book is written for high-school graduates who have had no previous experience with computers or BASIC programming, but it is useful even for advanced programers and teachers.

The book is more than simply a BASIC text-book. It provides information about how to define problems and design algorithms in order to solve them efficiently.

The book consists of 13 chapters. The first 3 chapter provides a textual and pictorial introduction to computers and programming. Fundamental BASIC statements and structures are described in the following five sections. Chapters 9 through 13 can be used to provide information about data file processing, printed output designing, arrays and special matrix statements. Each section contains some exercises and useful self-assessment tests. The author emphasizes the advantages of structured programming and the efficiency of modular design. Indeed, every program example is designed and developed in a structured way.

The author's aim was to give information about the fundamentals of modern problem-solving, such as the stepwise refinement procedure for designing algorithms, modularity, etc.

An other advantage of the book is the choice of the form of BASIC language. The Microsoft BASIC used in the book is a widespread form of BASIC language.

L. Czinkóczki (Szeged)

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

Information for authors

Acta Cybernetica publishes only original papers in the field of computer sciences mainly in English, but also in French, German or Russian. Authors should submit two copies of manuscripts to the Editorial Board. The manuscript must be typed double-spaced on one side of the paper only. Footnotes should be avoided and the number of figures should be as small as possible. For the form of references, see one of the articles previously published in the journal. A list of special symbols used in the manuscript should be supplied by the authors.

A galley proof will be sent to the authors. The first-named author will receive 50 reprints free of charge.

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1985. május 16.
Megjelenés: 1986. február
Terjedelem: 10,15 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
85-2217 — Szegedi Nyomda — F.v.: Dobó József igazgató