M. DEMLOVÁ
J. DEMEL
V. KOUBEK

On subdirectly irreducible automata

RAIRO – Informatique théorique, tome 15, nº 1 (1981), p. 23-46.
<http://www.numdam.org/item?id=ITA_1981__15_1_23_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 15, n° 1, 1981, p. 23 à 46)

ON SUBDIRECTLY IRREDUCIBLE AUTOMATA (*)

by M. Demlová (1), J. Demel (2) and V. Koubek (3)

Communicated by J. F. Perrot

Abstract. — *An automaton (finite or infinite) is* subdirectly irreducible *iff it has no non-trivial parallel decomposition. An algebraic characterization of such automata is given as well as an algorithm for deciding whether a given finite automaton is subdirectly irreducible that runs in polynomial time.*

Résumé. — *Les automates sous-directement irréductibles sont ceux qui n'admettent point de décomposition en parallèle non-triviale. On donne une caractérisation algébrique de ces automates ainsi qu'un algorithme pour décider si un automate fini donné est sous-directement irréductible; cet algorithme a une complexité polynomiale en temps.*

The structure theory of automata developed by Hartmanis and Stearns [8] deals with the decomposition of a given automaton into a network of automata. One of the basic types of decomposition is the parallel decomposition. The aim of the present paper is to characterize those automata which have no non-trivial parallel decomposition. More precisely, to characterize automata M with the following property: if M is a subautomaton of a parallel composition of automata $M_i, i \in I$, then there exists $i_0 \in I$ such that M is a subautomaton of M_{i_0}. Such automata we call *subdirectly irreducible* since similar objects in algebra are so called, *see* e. g. [3].

This paper is divided into two parts; the first gives an algebraic characterization of subdirectly irreducible automata (including infinite ones). The algebraic characterization is a generalization of a well-known description of

(*) Received July 1978, revised November 1979.
(1) Katedra Matematik, FEL ČVUT, Suchbâtarova 2, 16627 Praha 6, Czechoslovakia.
(2) Katedra ASŘ, FSv ČVUT, Thákurova 7, 16629 Praha 6, Czechoslovakia.
(3) VCUK, MFF UK, Malostranské nám. 25, 11800 Praha 1, Czechoslovakia.

subdirectly irreducible group automata, *see* e. g. Hartmanis and Stearns [8], Clifford and Preston [5]. The characterization for Medvedev automata is a direct continuation of an investigation of Schein [9] and Tully [12].

The second part is devoted to algorithms which decide whether a given finite automaton is subdirectly irreducible. For Medvedev automata and Mealy or Moore automata with one output, the algorithms work in the time proportional to $n^2(m+\log n)$, where $n = \text{card}\, Q$, $m = \text{card}\, X$, Q is the set of states, X is the set of inputs. The algorithms for Mealy and Moore automata with two outputs use Hopcroft's algorithm for minimizing the number of states in a finite automaton and thus they need a time proportional to $n.m.\log n$.

As a consequence of these investigations we get a description of subdirectly irreducible transformation semigroups and algorithms for deciding the irreducibility of finite ones.

Through the whole paper X, Q, Y resp. denotes the set of inputs, states, outputs resp. δ denotes the transition function $\delta : Q \times X \to Q$, β is the output function. A *Medvedev automaton* is a triple (X, Q, δ), a *Moore automaton* is a quintuple (X, Y, Q, δ, β), where $\beta : Q \to Y$, and a *Mealy automaton* is a quintuple (X, Y, Q, δ, β), where $\beta : Q \times X \to Y$.

DEFINITION: Let M, M' be automata (Medvedev, Moore, Mealy). We say that M' is a *subautomaton* of M if there exists:

for the Medvedev automata a couple (h_1, h_2),

for the Moore and Mealy automata a triple (h_1, h_2, h_3)

of one-to-one mappings $h_1 : X \to X'$, $h_2 : Q \to Q'$, $h_3 : Y \to Y'$, such that the following diagrams commute:

Medvedev:

$$\begin{tikzcd}
Q \arrow[r, "h_2"] & Q' \\
Q \times X \arrow[u, "\delta"] \arrow[r, "h_2 \times h_1"] & Q' \times X' \arrow[u, "\delta'"']
\end{tikzcd}$$

Moore:

$$\begin{tikzcd}
Y \arrow[r, "h_3"] & Y' \\
Q \arrow[u, "\beta"] \arrow[r, "h_2"] & Q' \arrow[u, "\beta'"'] \\
Q \times X \arrow[u, "\delta"] \arrow[r, "h_2 \times h_1"] & Q' \times X' \arrow[u, "\delta'"']
\end{tikzcd}$$

Mealy:

$$\begin{tikzcd}
Y \arrow[r, "h_3"] & Y' \\
Q \times X \arrow[u] \arrow[d] \arrow[r, "h_2 \times h_1"] & Q' \times X' \arrow[u] \arrow[d] \\
Q \arrow[r, "h_2"] & Q'
\end{tikzcd}$$

Let us recall several basic notions; we formulate them only for Mealy automata and omit the analogous definitions for Moore and Medvedev automata.

DEFINITION: We say that a Mealy automaton M is *subdirectly irreducible* provided that, whenever M is a subautomaton of a parallel composition of Mealy automata M_i, $i \in I$, with the triple (h_1, h_2, h_3) then there exists $i_0 \in I$ such that M is a subautomaton of M_{i_0} with the triple $(\pi_{i_0}^X \circ h_1, \pi_{i_0}^Q \circ h_2, \pi_{i_0}^Y \circ h_3)$ (where e. g. $\pi_{i_0}^X$ is the projection from the parallel composition to X_{i_0}).

DEFINITION: Let M be a Mealy automaton. A triple $(\lambda, \mu, \varepsilon)$ is called a *congruence* on M if λ is an equivalence on X, μ on Y, ε on Q such that:

$$x \lambda x', \quad q \varepsilon q' \quad \Rightarrow \quad \delta(q, x) \varepsilon \delta(q', x') \quad \& \quad \beta(q, x) \mu \beta(q', x').$$

Note: For Medvedev automaton a congruence (λ, ε) for which λ is the identical equivalence is called in [8] a partition with the substitution property.

DEFINITION: A system of congruences $\{ (\lambda_i, \mu_i, \varepsilon_i); i \in I \}$ on a Mealy automaton M is *separative* if $\{ \lambda_i; i \in I \}$, $\{ \mu_i; i \in I \}$ and $\{ \varepsilon_i; i \in I \}$ are separative systems of equivalences on X, Y, Q. [A system of equivalences $\{ \tau_i; i \in I \}$ on a set A is separative if for every $a, b \in A$, $a \neq b$, there exists $j \in I$ with $(a, b) \notin \tau_j$.]

CONVENTION: We shall denote by Δ_A the least equivalence on a set A, i. e. $(a, b) \in \Delta_A$ iff $a = b$; ∇_A the largest equivalence, i. e. $(a, b) \in \nabla_A$ for every $a, b \in A$. Evidently, $(\Delta_X, \Delta_Y, \Delta_Q)$ and $(\Delta_X, \nabla_Y, \nabla_Q)$ are congruences for every Mealy automaton; we shall call $(\Delta_X, \Delta_Y, \Delta_Q)$ the *identical congruence*.

The following proposition is a well-known analogue to that given by Birkhoff [4], *see* also [3], for algebras. Therefore, we shall formulate it without a proof.

PROPOSITION 1.1: *A Mealy (Moore, Medvedev) automaton is subdirectly irreducible iff every separative system of congruences on M contains the identical congruence. Equivalently: the set of all non-identical congruences on M has a smallest element (in the lattice of all congruences).*

Before we exhibit a characterization of subdirectly irreducible automata we shall point out several necessary conditions which enable us to restrict ourselves to Medvedev automata of a special form.

LEMMA 1.2: *Every subdirectly irreducible automaton (Mealy or Moore) has at most two outputs.*

The proof immediately follows from the fact that for every equivalence λ on Y, $(\Delta_X, \lambda, \Delta_Q)$ is a congruence on M.

LEMMA 1.3: *Let M be a subdirectly irreducible Mealy (Moore, Medvedev resp.) automaton. Let there exist $x, x' \in X$, $x \neq x'$, such that for every $q \in Q$, $\delta(q, x)=\delta(q, x')$ and $\beta(q, x)=\beta(q, x')$ [$\delta(q, x)=\delta(q, x')$, resp.]. Then* card $X=2$, card $Q=1$, *and* card $Y \leqq 1$.

Proof: Given an automaton (Mealy, Moore or Medvedev), define an equivalence λ on X by $(x, x') \in \lambda$ iff for all $q \in Q$, $\delta(q, x)=\delta(q, x')$ and $\beta(q, x)=\beta(q, x')$. Clearly, $(\lambda, \Delta_Y, \Delta_Q)$ is a congruence. Since $\{(\lambda, \Delta_Y, \Delta_Q), (\Delta_X, \nabla_Y, \nabla_Q)\}$ is a separative system of congruences we get our assertion.

CONVENTION: In the following we shall always assume that every Mealy (Moore, Medvedev resp.) automaton satisfies:

$$\left.\begin{array}{l}\forall x, x' \in X \quad (x \neq x' \Rightarrow \exists q \in Q : \delta(q, x) \neq \delta(q, x') \text{ or } \beta(q, x) \neq \beta(q, x')) \\ \quad (\forall x, x' \in X (x \neq x' \Rightarrow \exists q \in Q : \delta(q, x) \neq \delta(q, x')) \text{ resp.}).\end{array}\right\} \qquad (2)$$

We shall use the well-known notations: $X^*, X^+, \delta^*, \delta^+, \beta^*, \beta^+$ (*see* e. g. [2]).

THEOREM 1.4: *Let M be a Mealy (Moore resp.) automaton with* card $Y=2$. *Then the following are equivalent:*

(i) *M is subdirectly irreducible;*

(ii) *M satisfies (2) and the following holds: for every $q, q' \in Q, q \neq q'$, there exists $a \in X^+$ ($a \in X^*$ resp.), such that $\beta^+(q, a) \neq \beta^+(q', a)$, ($\beta^*(q, a) \neq \beta^*(q', a)$* resp.).

Proof: (ii)$\Rightarrow$(i). Take a non-identical congruence $(\lambda, \mu, \varepsilon)$ different from $(\Delta_X, \nabla_Y, \Delta_Q)$. From (2) we know that $(\lambda, \mu, \varepsilon)$ is non-identical iff $(\Delta_X, \mu, \varepsilon)$ is. Take any $(q, q') \in \varepsilon$, $q \neq q'$; by condition (ii) there is $a \in X^+$, resp. $a \in X^*$, with $\{\beta^+(q, a), \beta^+(q', a)\}=Y$, resp. $\{\beta^*(q, a), \beta^*(q', a)\}=Y$, and thus $\mu=\nabla_Y$. Hence $(\Delta_X, \nabla_Y, \Delta_Q)$ is the finest non-identical congruence.

(i)$\Rightarrow$(ii). Put

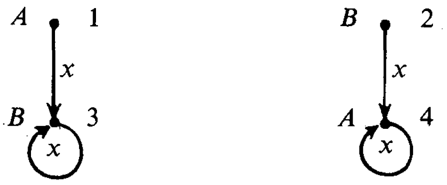
$$\varepsilon=\{(q, q');\ \forall a \in X^+ (\forall a \in X^* \text{ resp.}),$$
$$\beta^+(q, a)=\beta^+(q', a)(\beta^*(q, a)=\beta^*(q', a) \text{ resp.})\}.$$

If $(q, q') \in \varepsilon$ then $(\delta(q, x), \delta(q', x)) \in \varepsilon$ for every $x \in X$ [use e. g. $\beta^+(\delta(q, x), a)=\beta^+(q, xa)$]. Thus $(\Delta_X, \Delta_Y, \varepsilon)$ is a congruence on M. Clearly, $\{(\Delta_X, \Delta_Y, \varepsilon), (\Delta_X, \nabla_Y, \Delta_Q)\}$ is a separative system of congruences. Hence, if M is subdirectly irreducible, then $\varepsilon=\Delta_Q$, i. e. condition (ii) holds.

REMARK: Let $M=(X, Y, Q, \delta, \beta)$ be an automaton, card $Y=2$. Then subdirect irreducibility of M and of (X, Q, δ) are independent, i. e. there exists a subdirectly irreducible automaton M for which (X, Q, δ) is not subdirectly irreducible and on the other hand a non-subdirectly irreducible automaton M for which (X, Q, δ) is subdirectly irreducible.

Clearly, there exists e. g. a Moore automaton with two outputs which is not subdirectly irreducible, yet with (X, X, δ) subdirectly irreducible (for β being a constant mapping). The following example shows that there exists a subdirectly irreducible Moore automaton for which (X, Q, δ) is subdirectly reducible.

Example 1: Let $X=\{x\}$, $Y=\{A, B\}$, $Q=\{1, 2, 3, 4\}$. Let δ, β be defined as follows:



Clearly, (X, Q, δ) is not subdirectly irreducible, to see this it suffices to take $\varepsilon=\Delta_Q \cup \{(3, 4)\}$, $\varepsilon'=\Delta_Q \cup \{(1, 3)\}$ and we get a separative system of congruences $\{(\Delta_X, \varepsilon), (\Delta_X, \varepsilon')\}$. On the other hand, (X, Y, Q, δ, β) is subdirectly irreducible by theorem 1.4.

Using Theorem 1.4 and Lemma 1.2, we can restrict ourselves only to the case card $Y=1$ in the following. Clearly, a Moore or Mealy automaton with one output is subdirectly irreducible iff so is its Medvedev automaton. Thus from now on, throughout the first part, we shall consider only Medvedev automata.

Definition: Let M be an automaton. Define a graph $G=(Q, R)$ (i. e. $R \subseteq Q \times Q$) by: $(q, q') \in R$ if there exists $x \in X$ with $\delta(q, x)=q'$. The components of G (i. e. maximal connected subgraphs) will be called the *components of the automaton* M. If M has exactly one component we say that M is *weakly connected*.

Proposition 1.5: *Let M be a subdirectly irreducible automaton. Then either:* (1) *M is weakly connected or* (2) *M has two singleton components* (*and hence* card $X=1$) *or* (3) *M has two components, one of which is a singleton and for every element q in the other there exists $x \in X$ with $\delta(q, x) \neq q$.*

Proof: Let $\{E_i; i \in I\}$ be system of components of M. For every $i \in I$ define an equivalence ε_i by $(q, q') \in \varepsilon_i$ iff either $q=q'$ or $q, q' \notin E_i$. Clearly, $(\Delta_X, \varepsilon_i)$ is a congruence on M and $\{(\Delta_X, \varepsilon_i); i \in I\}$ is a separative system of congruences on M. Therefore there exists a component, the complement of which has at most one point. Hence either card $I=1$ or card $I=2$ and for some $i \in I$, card $E_i=1$. Assume the latter, i. e. $E_i=\{q'\}$, and let there exist $q \in E_j, j \neq i$, with $\delta(q, x)=q$ for all $x \in X$. Take (λ, ε) the smallest congruence with $(q, q') \in \varepsilon$. Clearly, $\lambda=\Delta_X$ and $\{(\Delta_X, \varepsilon_i), (\Delta_X, \varepsilon)\}$ is a separative system of congruences, thus $\varepsilon_i=\Delta_Q$, i. e. card $E_j=1$. By (2), card $X=1$.

To characterize subdirectly irreducible automata we use the following algebraic representation which is analogous to that given by Schein, *see* [9].

DEFINITION: Let X be a set. Then a triple $(S, \{\tau_i;\, i\in I\}, \tau)$ is said to be a *representation system* if the following conditions hold:

(1) S is a semigroup with the set of generators X;

(2) τ_i is a right congruence on S for every $i\in I$;

(3) τ is an equivalence on $=\{(R, \tau_i);\, R$ is a class of $\tau_i;\, i\in I\}$, such that: if $((R, \tau_i), (R', \tau_j))\in\tau$ then either $R=R'$ or $i\neq j$ and for every $x\in X$, $((Rx, \tau_i), (R', x, \tau_j))\in\tau$. ($Rx$ is the set of all elements $y.x$, $y\in R$, where . is the multiplication given by S.)

DEFINITION: Let M be an automaton. Then a representation system $(S, \{\tau_i;\, i\in I\}, \tau)$ over X is a *representation* of M if there exists a bijection $h : Q\to \mathbb{P}/\tau$ such that $h(\delta(q, x))$ is a class of τ containing (Rx, τ_i) whenever the class of $h(q)$ contains (R, τ_i). Define $\varphi : \mathbb{P}\to Q$ by $\varphi(R, \tau_i)= h^{-1}([R, \tau_i)])$. We call φ the *representation mapping* of $(S, \{\tau_i;\, i\in I\}, \tau)$ relative to M.

By a modification of the proof given in [9] we get the following proposition:

PROPOSITION 1.6: *Every automaton has a representation.*

There is a correspondence between Medvedev automata and transformation semigroups (shortly T-semigroups). Using this correspondence one obtains a representation of T-semigroups which is a generalization of the known representation of T-groups [5].

DEFINITION: A *T-semigroup* is a couple $(Z, \mathcal{M})$, where Z is a set and $\mathcal{M}$ a set of transformations of Z closed under composition. If all transformations are bijections and for every $f\in\mathcal{M}$ also $f^{-1}\in\mathcal{M}$, then $(Z, \mathcal{M})$ is called a *T-group*.

With every T-semigroup $(Z, \mathcal{M})$ we can associate a (Medvedev) automaton $A(Z, \mathcal{M})$ with $X=\mathcal{M}$, $Q=Z$, and $\delta(z, f)=f(z)$. Say that a representation system is a *representation of* $(Z, \mathcal{M})$ if it is a representation of $A(Z, \mathcal{M})$.

COROLLARY 1.7: *Every T-semigroup $(Z, \mathcal{M})$ has a representation. Moreover, there exists a representation $(S, \{\tau_j;\, j\in I\}, \tau)$ of $(Z, \mathcal{M})$, where $S=(\mathcal{M}, .)$ and if $(Z, \mathcal{M})$ is a T-group, then τ is identical.*

DEFINITION: A T-semigroup $(Z, \mathcal{M})$ has a *source* $x\in Z$ if for every $z\in Z$ there exists $f\in\mathcal{M}$ for which $f(x)=z$.

Let us give a characterization of several well-known notions (given e. g. in [2] and [8]) by means of the foregoing representation. The part of the following corollary concerning T-semigroups was first proved by Schein [9].

COROLLARY 1.8: (1) *An automaton M is:*

(*a*) *connected iff it has a representation* $(X, \{\tau_1\}, \tau)$;

(*b*) *strongly connected iff for each of its representations* $(S, \{\tau_i;\ i\in I\}, \tau)$ *the following holds: for arbitrary* $i\in I$, $a, b\in S$, *there exists* $c\in S$ *with* $(a, b.c)\in\tau_i$.

(2) *A T-semigroup* $(Z, \mathcal{M})$ *containing the identity mapping of* Z:

(*a*) *has a source* x *iff there exists a right congruence* $\bar{\tau}$ *on* $(\mathcal{M}, .)$, *for which* $(f(x), x)\in\bar{\tau}$ *whenever* $f\in\mathcal{M}$, *such that* $((\mathcal{M}, .), \{\bar{\tau}\}, \Delta_{\mathbb{P}})$ *is a representation of* $(Z, \mathcal{M})$ ($\mathbb{P}$ *is the set of classes of* $\bar{\tau}$);

(*b*) *is transitive iff for each of its representations* $(S, \{\tau_i;\ i\in I\}, \tau)$ *and for every* $i\in I$, $a, b\in S$, *there exists* $c\in S$ *with* $(a, b.c)\in\tau_i$.

CONVENTION: Let M be an automaton, $(S, \{\tau_i;\ i\in I\}, \tau)$ its representation. Denote by $\mathbb{C}_i$ the class of all right congruences on S strictly coarser than τ_i (note that $\tau_i\notin\mathbb{C}_i$).

DEFINITION: Let M be an automaton, $(S, \{\tau_i;\ i\in I\}, \tau)$ its representation. Let $(S, \{\mu_i;\ i\in I\}, \mu)$ be a representation system over X. We say that it is *compatible* with $(S, \{\tau_i;\ i\in I\}, \tau)$ if:

(1) $\mu_i\notin\mathbb{C}_i\cup\{\tau_i\}$ for every $i\in I$;

(2) if $((R, \tau_i), (R', \tau_j))\in\tau$ then $(([R]_{\mu_i}, \mu_i), ([R']_{\mu_j}, \mu_j))\in\mu$ (where $[R]_{\mu_i}$ is the class of μ_i containing R);

(3) if $(a, b)\in\mu_i$ and $(([a]_{\tau_i}, \tau_i), (R, \tau_j)), (([b]_{\tau_i}, \tau_i), (R', \tau_j))\in\tau$ then $[R]_{\mu_j}=[R']_{\mu_j}$.

LEMMA 1.9: *Let M be an automaton,* $(S, \{\tau_i;\ i\in I\}, \tau)$ *its representation. Let* $\lambda_i\in\mathbb{C}_i$. *Then there exists a compatible system* $(S, \{\lambda_j^i;\ j\in I\}, \lambda^i)$ *such that* $\lambda_i^i=\lambda_i$ *and every system* $(S, \{\nu_j;\ j\in I\}, \nu)$, *compatible with* $(S, \{\tau_j;\ j\in I\}, \tau)$ *for which* $\nu_i=\lambda_i$, *is compatible with* $(S, \{\lambda_j^i;\ j\in I\}, \lambda^i)$.

Proof: Define λ_j^i by $(a, b)\in\lambda_j^i$ iff either $(a, b)\in\tau_j$ or $(([a]_{\tau_j}, \tau_j), ([c]_{\tau_i}, \tau_i))$, $(([b]_{\tau_j}, \tau_j), ([d]_{\tau_i}, \tau_i))\in\tau$ and $(c, d)\in\lambda_i$. Define λ^i by $(([a]_{\lambda_j^i}, \lambda_j^i), ([b]_{\lambda_k^i}, \lambda_k^i))\in\lambda^i$ if there exist c, d with $(a, c)\in\lambda_j^i$, $(b, d)\in\lambda_k^i$ and $(([c]_{\tau_j}, \tau_j), ([d]_{\tau_k}, \tau_k))\in\tau$. Evidently, $(S, \{\lambda_j^i;\ j\in I\}, \lambda^i)$ has the required properties.

DEFINITION: Let $G=(V, E)$ be a graph. The *trap* $\mathcal{T}(G)$ of the graph G is defined to be the full subgraph of G on the set $T(G)\subset V$, being the least subset with the following properties:

(i) for every $a\in V$ there exists $b\in T(G)$ for which there is a (directed) path from a to b;

(ii) if $(a, b)\in E$ and $a\in T(G)$, then $b\in T(G)$.

Note. (1) Such $T(G)$ uniquely exists since sets with properties (i) and (ii) are closed under intersection.

(2) If there exists a (directed) path from a to b and $a, b \in T(G)$, then there also exists a (directed) path from b to a.

(3) If G' is a transitive closure of G, then $T(G) = T(G')$.

DEFINITION: Let M be an automaton. Denote by $G(M) = (P, R)$ the graph with $P = \{\{q, q'\}; q, q' \in Q, q \neq q'\}$ and, for every $p_1, p_2 \in P$, $(p_1, p_2) \in R$ iff for every congruence (λ, ε) for which $p_1 \in \varepsilon$ we have $p_2 \in \varepsilon$.

THEOREM 1.10: *Let M be an automaton fulfilling* (2). *Then the following are equivalent:*

(1) *M is subdirectly irreducible;*

(2) *the graph $\mathscr{T}(G(M))$ is complete;*

(3) *each representation $(S, \{\tau_i; i \in I\}, \tau)$ of M has the following properties:*

(*a*) *for every $i \in I$, $\mathbb{C}_i$ has the finest element* (*denote it by λ_i*),

(*b*) *for every $i, j \in I$, $\lambda^i_j \in \mathbb{C}_j$* (*for λ^i_j* see *Lemma* 1.9),

(*c*) *if $(S, \{\tau_i; i \in I\}, \tau')$ is a representation system then τ is coarser than τ'.*

Proof of (1)$\Leftrightarrow$(2): The graph $G(M)$ is transitive. Hence its trap $\mathscr{T}(G(M))$ consists of complete graphs with disjoint sets of vertices. Vertices of each of these complete graphs, say G_i, form a non-identical equivalence, say ε_i, on Q such that $(\Delta_X, \varepsilon_i)$ is a minimal congruence on M. (It follows from the definition of $G(M)$ and the trap.)

Now, M is subdirectly irreducible iff it has a finest non-identical congruence. This congruence is the only minimal non-identical congruence, i.e. $\mathscr{T}(G(M))$ has exactly one component. Using the transitivity of $G(M)$, this holds iff $\mathscr{T}(G(M))$ is complete.

Before proving (1)$\Leftrightarrow$(3) let us point out the following:

LEMMA 1.11: *Let M be an automaton with a representation $(S, \{\tau_i; i \in I\}, \tau)$. Let ε be an equivalence on Q. Then (Δ_X, ε) is a congruence on M iff there exists a system $(S, \{\mu_i; i \in I\}, \mu)$ compatible with $(S, \{\tau_i; i \in I\}, \tau)$ such that*

and
$$\left.\begin{aligned} (q, q') \in \varepsilon \quad &\Leftrightarrow \quad \exists\, a, b \in S, \quad i, j \in I, \\ \varphi([a]_{\tau_i}) = q, \qquad &\varphi([b]_{\tau_j}) = q' \\ (([a]_{\mu_i}, \mu_i), &([b]_{\mu_j}, \mu_j)) \in \mu, \end{aligned}\right\} \tag{3}$$

where φ is the representation mapping of $(S, \{\tau_i; i \in I\}, \tau)$ relative to M.

Proof: (1) Assume (Δ_X, ε) is a congruence. Then one can clearly construct a representation of $M/(\Delta_X, \varepsilon) = (X, Q/\varepsilon, \delta/\Delta_X \times \varepsilon)$ with the required properties.

(2) Let $(S, \{\mu_i; i \in I\}, \mu)$ be a representation system fulfilling (3). Take $(q, q') \in \varepsilon$, $x \in X$; then $\varphi([a]_{\tau_i}) = q$, $\varphi([b]_{\tau_j}) = q'$ for some $i, j \in I$, $a, b \in S$, and

$$(([a]_{\mu_i}, \mu_i), ([b]_{\mu_j}, \mu_j)) \in \mu,$$

so is

$$(([ax]_{\mu_i}, \mu_i), ([bx]_{\mu_j}, \mu_j)) \in \mu.$$

Further $\delta(q, x) = \varphi([ax]_{\tau_i})$, $\delta(q', x) = \varphi([bx]_{\tau_j})$, hence $(\delta(q, x), \delta(q', x)) \in \varepsilon$ and (Δ_X, ε) is a congruence.

CONVENTION: For every congruence (Δ_X, ε) denote

$$\Psi(\varepsilon) = (S, \{\mu_i; i \in I\}, \mu)$$

(*see* Lemma 1.11). Now, Ψ is a one-to-one correspondence between congruences on M containing Δ_X and systems compatible with $(S, \{\tau_i; i \in I\}, \tau)$ satisfying (3). Denote $\Phi = \Psi^{-1}$.

Proof of Theorem 1.10 (1) $\Rightarrow$ (3): Assume card $Q \neq 1$. (For card $Q = 1$ there is nothing to prove.)

Condition 3 (*a*): Assume that $\mathbb{C}_i$ does not contain a finest element, so there are $\sigma_i, \nu_i \in \mathbb{C}_i$ the intersection of which is τ_i. Thus we have two congruences (Δ_X, η), (Δ_X, ξ) on M, where

$$\eta = \Phi(S, \{\sigma_j^i; j \in I\}, \sigma^i), \qquad \xi = \Phi(S, \{\nu_j^i; j \in I\}, \nu^i)$$

(*see* Lemma 1.9 and the above convention). Let $(q, q') \in \eta$, $q \neq q'$, i.e. by Lemma 1.9 and Lemma 1.11 there exist $a, b \in S$ with $(a, b) \in \sigma_i$ and $\varphi([a]_{\sigma_i}) = q$, $\varphi([b]_{\sigma_i}) = q'$. Since $(a, b) \notin \tau_i$, we have $(a, b) \notin \nu_i$ and thus $(q, q') \notin \xi$. Analogously, one proves that for every $(q, q') \in \xi$, $q \neq q'$, we have $(q, q') \notin \eta$. Hence (Δ_X, η), (Δ_X, ξ) form a separative system of congruences, i.e. M is not subdirectly irreducible.

Condition 3 (*b*): Assume that there exist $\mu_i \in \mathbb{C}_i$ and $j \in I$ with $\mu_j^i = \tau_j$. Then we can assume that $\mu_i = \lambda_i$. Take $\nu = \Phi(S, \{\lambda_k^i; k \in I\}, \lambda^i)$ and $\sigma = \Phi(S, \{\lambda_k^j; k \in I\}, \lambda^j)$. Clearly (Δ_X, ν), (Δ_X, σ) form a separative system of congruences, hence M is not subdirectly irreducible.

Condition 3 (*c*): Assume that there exists an equivalence τ' on $\mathbb{P}$ coarser than τ such that $(S, \{\tau_i; i \in I\}, \tau')$ is a representation system satisfying (3). Take $\sigma = \Phi(S, \{\tau_i; i \in I\}, \tau')$, $\nu = \Phi(S, \{\lambda_k^j; k \in I\}, \lambda^j)$ for some $j \in I$. Again (Δ_X, σ), (Δ_X, ν) form a separative system of congruences. Indeed, if $(p, r) \in \sigma$, then for $j \in I$ it holds: if $\varphi([a]_{\tau_j}) = p$ for some $a \in S$, then for every $b \in S$, $\varphi([b]_{\tau_j}) \neq r$, hence by

Lemma 1.9, $(p, r) \notin \nu$. If $(p, r) \in \nu$ then there exist $a, b \in S$ with $\varphi([a]_{\tau_j}) = p$, $\varphi([b]_{\tau_j}) = r$ and $(a, b) \in \lambda_j$ but $(a, b) \notin \tau_j$. Thus $(p, r) \notin \sigma$ and the proof is complete.

(3) $\Rightarrow$ (1). Let $\mathscr{S} = \{(\sigma_k, \mu_k); k \in J\}$ be a separative system of congruences on M. One has: (Δ_X, μ) is a congruence on M whenever there exists σ, with (σ, μ) being a congruence. So we can suppose that $\sigma_k = \Delta_X$ for all $k \in J$. Take

$$\{\Psi(\mu_k); k \in J\}, \quad \Psi(\mu_k) = (S, \{\bar{\mu}_i^k; i \in I\}, \bar{\mu}^k).$$

By Condition 3 (c) we have: whenever $\mu_k \neq \Delta_Q$ there exists i_0 with $\bar{\mu}_k^{i_0} \in \mathbb{C}_{i_0}$. Hence by Condition 3 ($a$) (Δ_X, μ_k) is coarser than $\Phi(S, \{\lambda_j^{i_0}, j \in I\}, \lambda^{i_0})$. Further from Condition 3 (b) we have that $\lambda_j^i \supseteq \lambda_j$ for every $i, j \in I$. Thus, from the separativity of $\mathscr{S}$, there exists $k_0 \in J$ for which $\mu_{k_0} = \Delta_Q$. By (2), $\sigma_{k_0} = \Delta_X$ and thus $(\Delta_X, \Delta_Q) \in \mathscr{S}$ and M is subdirectly irreducible.

DEFINITION: A T-semigroup $(Z, \mathscr{M})$ is a *T-subsemigroup* of $(Y, \mathscr{N})$ if there exists a pair (f, η) of one-to-one mappings $f: Z \to Y$, $\eta: \mathscr{M} \to \mathscr{N}$ such that:

(1) $\eta(g_1 \circ g_2) = \eta(g_1) \circ \eta(g_2)$ for every $g_1, g_2 \in \mathscr{M}$;

(2) $f(g(x)) = \eta(g)(f(x))$ for every $x \in Z$, $g \in \mathscr{M}$.

Then (f, η) is the *inclusion morphism* of $(Z, \mathscr{M})$ into $(Y, \mathscr{N})$.

A T-semigroup $(Z, \mathscr{M})$ is *subdirectly irreducible* if: whenever $(Z, \mathscr{M})$ is a T-subsemigroup of $(\prod_{i \in I} Z_i, \{\prod_{i \in I} g_i; g_i \in \mathscr{M}_i\})$ ($(Z_i, \mathscr{M}_i)$ are T-semigroups) with the inclusion morphism (f, η), then there exists $i_0 \in I$ such that $(Z, \mathscr{M})$ is a T-subsemigroup of $(Z_{i_0}, \mathscr{M}_{i_0})$ with $(\pi_{i_0} \circ f, \pi_{i_0} \circ \eta)$ being the inclusion morphism. (π_i means the i-th projection of the product.)

COROLLARY 1.12: *For a T-semigroup $(Z, \mathscr{M})$ the following are equivalent:*

(1) *$(Z, \mathscr{M})$ is subdirectly irreducible;*

(2) *the trap of the graph*

$$(\{\{x, y\}; x, y \in Z, x \neq y\}; \{(\{x, y\}, \{z, v\});$$
$$\{x, y\} \neq \{z, v\}, \forall \text{ congruence } \varepsilon \text{ on } (Z, \mathscr{M}) ((x, y) \in \varepsilon \Rightarrow (z, v) \in \varepsilon)\})$$

is complete;

(3) *every representation of $(Z, \mathscr{M})$ satisfies Condition 3 from Theorem 1.10.*

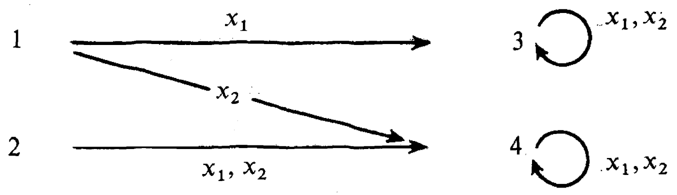
COROLLARY 1.13: *If an automaton M is connected (a T-semigroup $(Z, \mathscr{M})$ has a source) then M ($(Z, \mathscr{M})$) is subdirectly irreducible iff there exists a representation $(S, \{\mathscr{T}\}, \Delta_{\mathbb{P}})$ such that $\mathbb{C}_\tau$ has a finest element. Moreover, let $(Z, \mathscr{M})$ be a T-group. Then $(Z, \mathscr{M})$ is subdirectly irreducible iff for some $x \in Z$ the set $\{G; G$ is a subgroup of $(\mathscr{M}, .)$, $\{f \in \mathscr{M}; f(x) = x\} \subsetneqq G\}$ has a smallest element.*

Note: The last statement is a well-known description of subdirectly irreducible T-groups.

Note: Condition 3 from Theorem 1.10 is equivalent to the following: there exists a representation of M with 3 (*a*)-3 (*c*).

REMARK: Neither Condition 3 (*b*) nor 3 (*c*) can be ommited, as can be seen from the following examples.

Example 2: Let M be an automaton, where $X=\{x_1, x_2\}$, $Q=\{1, 2, 3, 4\}$ and

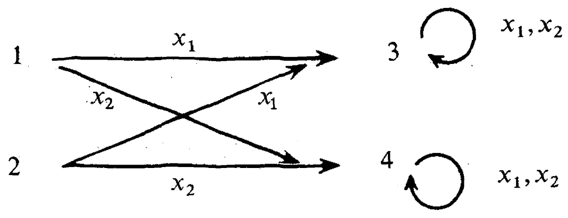


A representation of M: Put $S=X^*$, take $I=\{1, 2\}$; τ_1 has three classes: $\{\Lambda\}$, $\{a; \exists\, b\in X^*,\ a=x_1 b\}$, $\{a; \exists\, b\in X^*,\ a=x_2 b\}$, τ_2 has two classes: $\{\Lambda\}$, $\{a; a\in X^*, a\neq\Lambda\}$ (Λ is the empty word). Further,

$$\mathbb{P}=\{([\Lambda]_{\tau_1}, \tau_1), ([x_1]_{\tau_1}, \tau_1), ([\Lambda]_{\tau_2}, \tau_2), ([x_1]_{\tau_2}, \tau_2), ([x_2]_{\tau_1}, \tau_1)\}.$$

τ is generated by $(([x_2]_{\tau_1}, \tau_1), ([x_1]_{\tau_2}, \tau_2))\in\tau$. Clearly, M satisfies (2) and Conditions 3 (*a*), 3 (*c*). But it does not satisfy 3 (*b*) since the finest congruence in $\mathbb{C}_2$ is $\lambda_2=\nabla_X$ and $\lambda_1^2=\tau_1$. Therefore M is not subdirectly irreducible.

Example 3: Let M be an automaton, where $X=\{x_1, x_2\}$, $Q=\{1, 2, 3, 4\}$ and



A representation of M: Put $S=X^*$. Take $I=\{1, 2\}$, $\tau_1=\tau_2$ and both have three classes: $\{\Lambda\}$, $\{a; a=x_1 b\}$, $\{a; a=x_2 b\}$.

$$\mathbb{P}=\{([\Lambda]_{\tau_i}, \tau_i), ([x_j]_{\tau_i}, \tau_i);\ i=1, 2, j=1, 2\}.$$

Further, τ is generated by $(([x_i]_{\tau_1}, \tau_1), ([x_i]_{\tau_2}, \tau_2))\in\tau$, $i=1, 2$. Evidently, M satisfies (2) and Conditions 3 (*a*), 3 (*b*). But it does not satisfy Condition 3 (*c*). For this take τ' which contains moreover $(([\Lambda]_{\tau_1}, \tau_1), ([\Lambda]_{\tau_2}, \tau_2))$. Thus M is not subdirectly irreducible.

II

This part exhibits algorithms for deciding whether a given finite automaton (Medvedev, Moore, Mealy) is subdirectly irreducible. For this purpose we shall use the following three algorithms $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{K}$.

ALGORITHM $\mathscr{A}$: For a given mapping $h : B \times A \to C$ algorithm $\mathscr{A}$ finds the equivalence α on A defined by $(a, a') \in \alpha$ iff $h(b, a) = h(b, a')$ for every $b \in B$. Moreover algorithm $\mathscr{A}$ decides whether $\alpha = \Delta_A$. The worst case running time of this algorithm is proportional to $\operatorname{card} A . \operatorname{card} B$.

ALGORITHM $\mathscr{B}$: For a given Medvedev automaton M algorithm $\mathscr{B}$ finds the trap $\mathscr{T}(G(M))$ and decides whether the trap is a complete graph. The worst case running time is proportional to $n^2.(m + \log n)$, where $n = \operatorname{card} Q$, $m = \operatorname{card} X$.

A description of algorithms $\mathscr{A}$, $\mathscr{B}$, a proof of their correctness and a proof of the running time is given below.

ALGORITHM $\mathscr{K}$: For a given Moore or Mealy automaton with the set of states Q and for a given equivalence ε on Q algorithm $\mathscr{B}$ finds the biggest congruence $(\Delta_X, \Delta_Y, \lambda)$ such that λ is finer than ε. Moreover it decides whether $\lambda = \Delta_Q$. The worst case running time is proportional to $n.m.\log n$.

This algorithm can be obtained as a slight modification of Hopcroft's algorithm for minimizing the number of states in a finite automaton (*see* [1]).

Now, we shall exhibit the algorithms for deciding whether a given automaton is subdirectly irreducible. We do not deal with the trivial case $\operatorname{card} Q = \operatorname{card} Y = 1$, $\operatorname{card} X \leq 2$; such an automaton is subdirectly irreducible.

Medvedev automaton M

First we decide whether Condition (2) holds. For this we use algorithm $\mathscr{A}$ for $A = X$, $B = C = Q$ and $h = \delta$. If $\alpha \neq \Delta_X$, the automaton is not subdirectly irreducible. If $\alpha = \Delta_X$, then we use algorithm $\mathscr{B}$ for deciding whether the trap $\mathscr{T}(G(M))$ is complete. If $\mathscr{T}(G(M))$ is complete, the automaton M is subdirectly irreducible, otherwise M is not subdirectly irreducible.

The time bound is proportional to $n^2(m + \log n)$, where $n = \operatorname{card} Q$, $m = \operatorname{card} X$.

Moore automaton M

If $\operatorname{card} Y > 2$, then M is not subdirectly irreducible. If $\operatorname{card} Y = 1$, then M is subdirectly irreducible iff (X, Q, δ) is so; thus we use the above algorithm.

If card $Y=2$, then M satisfies (2) iff so does (X, Q, δ); thus we use algorithm $\mathscr{A}$ in the same way as above. If M satisfies (2) then we use algorithm $\mathscr{X}$ for the equivalence γ defined by: $(q, q')\in\gamma$ iff $\beta(q)=\beta(q')$ (as in the original Hopcroft's algorithm). If $\lambda=\Delta_Q$, then M is subdirectly irreducible, otherwise M is not subdirectly irreducible.

For card $Y=2$ the time bound is proportional to $n.m.\log n$.

Mealy automaton M

If card $Y\neq 2$, then the algorithm is the same as for a Moore automaton.

If card $Y=2$, then in order to decide whether M fulfills (2) we use algorithm $\mathscr{A}$ for $A=X$, $B=Q$, $C=Q\times Y$ and $h=\delta\times\beta$ [i.e. $h(q, x)=(\delta(q, x), \beta(q, x))$].

If M satisfies (2), then we use algorithm $\mathscr{X}$ for the equivalence γ defined by $(q, q')\in\gamma$ iff, for every $x\in X$, $\beta(q, x)=\beta(q', x)$. To obtain γ we make use of algorithm $\mathscr{A}$, now for $A=Q$, $B=X$, $C=Y$ and $h(x, q)=\beta(q, x)$. Automaton M is subdirectly irreducible iff the equivalence λ obtained by algorithm $\mathscr{X}$ is equal to Δ_Q.

For card $Y=2$ the time bound is again proportional to $n.m.\log n$.

Algorithm $\mathscr{A}$

Let A, B, C be finite sets, $h: B\times A\to C$ a mapping.

DEFINITION: Let $B'\subseteq B$. We say that a, a' are *recognizable w.r.t. B'* provided there exists $b\in B'$ with $h(b, a)\neq h(b, a')$. Further, define an equivalence $\sim_{B'}$ on A putting $a\sim_{B'}a'$ iff a, a' are not recognizable w.r.t. B'. The system of classes of $\sim_{B'}$ is called the *partition w.r.t. B'*.

Our algorithm constructs a sequence of partitions $\mathscr{P}_k$ w.r.t. $B_k=\{b_1, \ldots, b_k\}$, $k=1, \ldots, n=\text{card}\,B$. First, let us describe the procedure which constructs partition $\mathscr{P}_{k+1}$ w.r.t. $B_k\cup\{b_{k+1}\}$ from the partition $\mathscr{P}_k$. Every block P_k^i will be split into $\{\pi_2(h^{-1}(c)); c\in h(b_{k+1}, P_k^i)\}$, where π_2 is the projection $\pi_2(b, a)=a$. This can be achieved in a time proportional to card P_k^i. Hence the time needed for the construction of $\mathscr{P}_{k+1}$ is proportional to card A.

LEMMA 2.1: *For $k=1, \ldots, n-1$, with $n=\text{card}\,B$, $\mathscr{P}_{k+1}$ is a refinement of $\mathscr{P}_k$ and $\mathscr{P}_n$ corresponds to the required equivalence α.*

Description of algorithm $\mathscr{A}$

We use $(n+1)$-lists M_i, $i=1, \ldots, n+1$; every M_i is either empty or contains a block of $\mathscr{P}_k$ or contains a block of $\mathscr{P}_{k+1}$. At every moment the set of all indices of lists M_i is divided into four parts: L_E the set of indices of M_i which are empty, L_o

the set of indices of M_i containing blocks of $\mathscr{P}_k$ which are to be split w. r. t. b_{k+1}, L_N the set of indices of M_i in which all blocks of $\mathscr{P}_{k+1}$ have at least two elements, L_1 the set of indices of M_i which contain singleton blocks.

When splitting block P_k^j the symbol $t(c)$ means the index i such that M_i will contain block P_{k+1}^r for which $h(b_{k+1}, P_{k+1}^r)=\{c\}$ provided such an index i exists, otherwise $t(c)=0$ ($0 \notin A \cup B \cup C$).

```
Initialize: $M_1=A$, $M_i=\emptyset$ for $i=2, \ldots, n+1$
            $t(c)=0$ for all $c \in C$
            $L_N = L_1 = \emptyset$
            $L_0 = \{1\}$, $L_E=\{2, \ldots, n+1\}$
a: for each $b \in B$ do
b :     while $L_0 \neq \emptyset$ do
            take $i \in L_0$;
c: (Splitting $M_i$ w. r. t. $b$):
        while $M_i \neq \emptyset$ do
            take $a \in M_i$;
            $M_i=M_i \setminus \{a\}$;
            $c=h(b, a)$;
            if $t(c)=0$ then begin;
                        take $j \in L_E$;
                        $L_E=L_E \setminus \{j\}$;
                        $L_N=L_N \cup \{j\}$;
                        $t(c)=j$;
                    end; $M_j=\{a\}$;
                    else $M_{t(c)}=M_{t(c)} \cup \{a\}$;
            end c;
            $L_0=L_0 \setminus \{i\}$;
            $L_E=L_E \cup \{i\}$;
        end b;
d:      while $L_N \neq \emptyset$ do
            take $i \in L_N$;
            $L_N=L_N \setminus \{i\}$;
            if card $M_i=1$ then $L_1=L_1 \cup \{i\}$;
                           else $L_0=L_0 \cup \{i\}$;
        end d;
        if $L_0=\emptyset$ then begin; write "$\alpha=\Delta_A$"; stop; end;
                    else;
    end a;
    write "$\alpha \neq \Delta_A$"; stop;
```

Correctness of algorithm $\mathscr{A}$ follows from Lemma 2.1. It remains to show that the number of lists M_i is sufficient. But this follows from the following observation: Let M_i contain k elements. Since every $a \in A$ is in every moment contained in at most one list, there are at most $n-k+1$ non-empty lists, i. e. there are at least k empty lists for splitting M_i.

The time needed for one execution of Statement c is proportional to card M_i. Hence the time needed for one execution of Statement b is proportional to card A. Statement d needs a time proportional to card $L_N \leqq$ card A. Both Statements b and d are repeated at most card B times, thus the time of execution of the whole algorithm is proportional to card A.card B.

Algorithm $\mathscr{B}$

Let M be a finite Medvedev automaton. Our aim is to give an algorithm that finds the trap $\mathscr{T}$ $G(M))$ and decides whether the trap is complete.

First, let us point out several lemmas about finding the trap of a (general) graph.

DEFINITION: Let $G=(V, E)$ be a graph. Define an equivalence $c(G)$ on V as follows: $(u, v) \in c(G)$ iff there exist paths both from u to v and from v to u. Denote by $C(G)$ the quotient graph $(V/c(G), E/c(G))$ without loops, i. e. for different classes U_1, U_2 of $c(G)$, (U_1, U_2) is in $E/c(G)$ iff there exist $u_1 \in U_1$, $u_2 \in U_2$ with $(u_1, u_2) \in E$. The classes of $c(G)$ will be called *strongly connected components* of G.

Let us consider a sequence of graphs $G_0, \ldots, G_n$ such that :
(S1) all $G_i=(V, E_i)$ are subgraphs of G and $G_0=(V, \emptyset)$;
(S2) for every i, $E_{i+1}=E_i \cup \{(u, v)\}$ for some u, v such that $u \in T(G_i)$ and $(u, v) \notin c(G_i)$;
(S3) if $u \in T(G_n)$ and $(u, v) \in E$ then $(u, v) \in c(G_n)$.
Note that such a sequence exists for each graph.

LEMMA 2.2: (1) $T(G_n)=T(G)$.
(2) $c(\mathscr{T}(G_n))=c(\mathscr{T}(G))$.

Note: $u \in T(G)$ iff for every path from u to v there exists a path from v to u.

The proof of Lemma 2.2 follows from the above note and from the fact that by addition of an edge $(u, v) \in E \setminus E_n$ to the graph G_n, both the set $T(G_n)$ and the equivalence $c(\mathscr{T}(G_n))$ do not change.

LEMMA 2.3: *For every* $i=0, \ldots, n$ *the graph* $C(G_i)$ *is a partial mapping without cycles.*

Note: In other words $C(G_i)$ is a forest.

Proof: Let us proceed by induction. Evidently, our assertion holds for G_0. Assume that it holds for G_i. By the definition $G_{i+1}=(V, E_{i+1})$, where $E_{i+1}=E_i \cup \{(u, v)\}$ for $u \in T(G_i)$. The following two cases can happen:
1) the grapn $C(G_i)$ with the added edge $([u]_i, [v]_i)$ has no cycle ($[u]_i$ denotes the element of $V/c(G_i)$ containing u). In this case the equivalence $c(G_i)$ and $c(G_{i+1})$ coincide and thus $C(G_{i+1})$ has no cycle;

2) the graph $C(G_i)$ with the added edge $([u]_i, [v]_i)$ contains a cycle, say $[v_1]_i$, $[v_2]_i, \ldots, [v_k]_i$. Then by the definition of $c(G_{i+1})$ the set $\bigcup_{j=1}^{k} [v_j]_i$ forms a strongly connected component of G_{i+1} and hence $C(G_{i+1})$ has no cycle.

Now, we shall show that $C(G_{i+1})$ is a partial mapping. Assume that, by way of contradiction, there exist X, Y, Z, $X \neq Y \neq Z \neq X$, such that both (X, Y) and (X, Z) are edges of $C(G_{i+1})$. Neither (X, Y), nor (X, Z) is the new added edge, since $X \nsubseteq T(G_i)$. Then there exist vertices D_1, D_2 of $C(G_i)$ such that $D_1 \cup D_2 \subseteq X$ and (D_1, Y), (D_2, Z) are edges of $C(G_i)$. Then there exists a path in G_i either from D_1 to D_2 or from D_2 to D_1 and all vertices of this path belong to X. Let there exist e. g. a path from D_1 to D_2. Since $C(G_i)$ is a partial mapping, Y is the second vertex in the path from D_1 to D_2. Then $Y \subseteq X$ and hence $X = Y$ – a contradiction.

LEMMA 2.4: *Let $G=(V, E)$ be a graph, let the sequence $G_0, \ldots, G_n$ fulfil* (S1), (S2), (S3). *Let $k = \text{card } V$. Then $n \leqq 2k-2$.* (In other words: graph G_n has at most $2k-2$ edges.)

Proof: By induction on i we shall prove that if a strongly connected component of G_i has s elements, then it has at most $2s-2$ edges.

This is evident for G_0.

Let the statement hold for G_i, we shall prove it for G_{i+1}. The graph G_{i+1} arises from G_i by addition of, say, the edge (u, v). There are two possibilities:

a) the graph $C(G_i)$ with the added edge has no cycle. In this case $c(G_i) = c(G_{i+1})$ and there is no new edge inside the strongly connected components of G_{i+1};

b) the graph $C(G_i)$ with the added edge contains a cycle consisting of strongly connected components $[v_1]_i, [v_2]_i, \ldots, [v_m]_i$. Denote by $k_1, \ldots, k_m$ the numbers of vertices of $[v_1]_i, [v_2]_i, \ldots, [v_m]_i$. In all these strongly connected components at most $m-1$ edges belong to G_i, because $C(G_i)$ is a forest. The new class of $c(G_{i+1})$ is a union of strongly connected components $[v_1]_i, \ldots [v_m]_i$. The number of edges in the new strongly connected component is at most

$$m + \sum_{i=1}^{m} (2s_i - 2) = 2. \sum_{i=1}^{m} s_i - m.$$

Since $m \geqq 2$, the new strongly connected component having $s = \sum_{i=1}^{m} s_i$ vertices, has at most $2s-2$ edges.

Now, let $[v_1], \ldots, [v_m]$ be all the strongly connected components of G_n, let $k_1, \ldots, k_m$ be numbers of vertices in them and $k = \sum_{i=1}^{m} k_i$. Then there are at most $\sum_{i=1}^{m} (2k_i - 2) + m - 1 \leqq 2k - 2$ edges in G_n.

Lemmas 2.2-2.4 enable us to give an algorithm for finding the trap of an (general) graph G. We shall construct a sequence $G_0, \ldots, G_n$ fulfilling (S1)-(S3). While constructing it we maintain the set $T(G_i)$, the graph $C(G_i)$ and the equivalence $c(G_i)$.

When a new edge (u, v) is added to G_i, the following two cases can occur:

1) no cycle is closed, then we have to exclude $[u]_i$ from the trap and add a new edge to $C(G_i)$. There are no changes in the equivalence $c(G_i)$;

2) a new cycle $u, v, v_1, v_2, \ldots, v_k$ is closed. Then we have to form a new strongly connected component $[v]_{i+1}$ of G_{i+1} as the union of strongly connected components $[u]_i, [v]_i, [v_1]_i, \ldots, [v_k]_i$ of G_i and also make corresponding changes in $T(G_i)$ and $C(G_i)$.

Using suitable data structures it is possible to generate new edges in a total time proportional to the number of edges in G. In addition, we can also remove a strongly connected component of G_i from the trap in a fixed amount of time; and this needs be done at most once for every edge of G.

The time needed for maintenance of $T(G_i)$, $c(G_i)$ and $C(G_i)$ when the new edge closes a cycle is proportional to $k.\log k$, where $k = \text{card } V$, by the following lemma.

LEMMA 2.5: *Let $\{\mathcal{P}_i\}_{i=0}^{n}$ be a sequence of partitions on the set V such that $\mathcal{P}_{i+1}$ is obtained from $\mathcal{P}_i$ as a union of two blocks of $\mathcal{P}_i$. Let the time needed for computing the union of two blocks be equal to the cardinality of the smaller of them. Then the total time for obtaining $\cup_n$ from $\cup_0$ is not larger then $(n/2).\log_2 n$, where $n = \text{card } V$.*

Proof: Without loss of generality we can assume that $\mathcal{P}_0$ consists of singleton blocks only and $\mathcal{P}_n$ has only one block. (This is the worst case.) We shall prove by induction that the time needed for establishing a block of cardinality a is at most $(a/2).\log_2 a$.

For $\mathcal{P}_0$, this is obvious. Let the statement hold for two blocks with cardinalities $a \leqq b$. We have to prove that

$$\frac{a}{2}.\log_2 a + \frac{b}{2}.\log_2 b + a \leqq \frac{a+b}{2}.\log_2 (a+b).$$

Since $a.\log_2 a + a = a.\log_2 (2a)$, it is sufficient to show that

$$\frac{b}{2}.\log_2 b - \frac{a}{2}.\log_2 a \leqq \frac{a+b}{2}.\log_2 (a+b) - a.\log_2 (2a).$$

But this follows from the fact that the function

$$f(x) = \frac{x}{2}.\log_2 x$$

is increasing and convex.

To save the time needed for a decision whether a new cycle was closed in G_{i+1}, we shall label vertices of G by natural numbers and choose the new edges in a special order using these labels. The label of vertex v is denoted by $L(v)$. The rules for labelling vertices and for choosing the edges are the following:

(L1) In G_0, there are no labels.

(L2) If there exists an edge $(u, v) \in E$ which can be added to G_i according to (S1)-(S3) such that vertex u has a label, then we shall choose this edge and we shall label $L(v) = L(u)$, provided v has no label in G_i.

(L3) If for every edge $(u, v) \in E$ that can be added to G_i according to (S1)-(S3), u has no label, then we shall choose an arbitrary vertex u which has no label and put $L(u) = 1 + \max \{ L(w);\ w \in V \text{ and } w \text{ has a label} \}$.

Note: (L3) describes the introduction of a new label and (L2) the addition of a new edge.

Rules (L1)-(L3) imply that we shall prefer edges leading from the component of the trap, which has just been created by closing a cycle or to which the last edge leads. The set of vertices labelled by the same label forms a branch of the forest $C(G_i)$.

LEMMA 2.6: *If both u and v have labels and $L(u) = L(v)$, then there exists a path either from u to v or from v to u.*

Proof: Let us proceed by induction on i. The statement holds for G_0; let it hold for G_{i-1}. Let $E_i \setminus E_{i-1} = \{(w, z)\}$. There are three possibilities:

a) both w and z have already labels in G_{i-1};

b) neither w nor z has a label in G_{i-1};

c) only one of vertices w, z has a label in G_{i-1}.

The only interesting case is the third one. In this case w has a label and z has no label in G_{i-1}. Let now $L(u) = L(w) = L(z)$ in G_i. Since $w \in T(G_{i-1})$, there exists a path from u to w in G_{i-1} and then there exists a path from u to z in G_i.

LEMMA 2.7: *A cycle is closed by adding the edge (u, v) to G_i iff both u and v have already labels in G_i and $L(u) = L(v)$.*

The proof immediately follows from Lemma 2.6 and from the fact that if there is a path from w to z, then $L(w) \geqq L(z)$.

Note: Lemmas 2.2-2.7 enable us to construct an algorithm for finding the trap of a (general) graph the time bound of which is proportional to card E + card V. log (card V).

Let us come back to the case of automata. We have to find the trap of the graph $G(M) = (P, R)$ (*see* part I). The algorithm will be based on the concepts described above for graphs. We shall construct a sequence $G_0, \ldots, G_n$ fulfilling (S1)-(S3). The vertices will be labelled and the new edge will be chosen as described in (L1)-(L3). However, the set of all edges of $G(M)$ is rather large, since the relation R is transitive. We shall show that the choice of edges can be restricted to edges of two special types:

Let us consider the edge (p, p'), where $p=\{q_1, q_2\}, p'=\{q'_1, q'_2\}$. We say that this edge (p, p') is of type L provided there exists a letter $x \in X$ such that $\delta(\{q_1, q_2\}, x)=\{q'_1, q'_2\}$. On the other hand this edge (p, p') is said of type T w. r. t. graph G_i provided $(p, p') \in E_{i+1} \setminus E_i$, $q_2=q'_2$ and there exists $p_1=\{q_1, q'_1\}$ such that $(p, p_1) \in c(G_i)$.

LEMMA 2.8: *Let the sequence of graphs* $G_0, \ldots, G_n$ *fulfilling* (S1)-(S2) *and* (L1)-(L3) *be constructed using edges of types T and L only. Let the condition* (S3) *hold for edges of types T and L. Then* $T(G(M))=T(G_n)$ *and* $c(\mathscr{T}(G(M)))=c(\mathscr{T}(G_n))$.

Proof: Let the sequence $G_0, \ldots, G_n$ satisfy the assumptions of Lemma 2.8. Let $G(M)=(P, R)$, $G_n=(P, R_n)$. Every strongly connected component $A \subseteq T(G_n)$ of the trap $\mathscr{T}(G_n)$ forms a relation ε on Q in the following way: $(q_1, q_2) \in \varepsilon$ iff $p=\{q_1, q_2\} \in A$. Since Condition (S3) holds for edges of types T and L, the relation ε is an equivalence and (Δ_X, ε) is a congruence of the automaton M. Then there exists no edge $(p, p') \in R \setminus R_n$ such that $p \in T(G_n)$ and $(p, p') \notin c(G_n)$. Thus Condition (S3) holds for all edges of $G(M)$. Now, Lemma 2.2 concludes the proof.

All edges of type L are defined by the mapping $\delta : Q \times X \to Q$; there is at most one edge for every pair of states $q_1, q_2 \in Q$ and for every letter $x \in X$. Hence the search for these edges is easy.

The search for edges of type T is a little more difficult: We have to examine whether the relation formed by the strongly connected component of the trap is transitive. If it is not transitive, then we have found an edge of type T. If the relation is transitive, no edge of type T leading from the strongly connected component exists.

Before giving the full description of algorithm $\mathscr{B}$, let us describe the data structures used in it.

From every strongly connected component of G_i a representative vertex is chosen. All information concerning the strongly connected component is accessible by means of this representative.

$R(p)$ denotes the representative of the strongly connected component containing p;

$K(r)$	denotes the list of all vertices of the strongly connected component, the representative of which is r;
$H(p)$	denotes the set of letters for which we have not yet examined the edge (of type L) leading from p;
$D(r)$	denotes the set of vertices $p \in K(r)$ for which $H(p) \neq \emptyset$;
$L(p)$	denotes the label of the vertex p, provided that p has a label. Otherwise $L(p)=0$;
l	denotes the maximum of all labels in G_i;
F	denotes the set of unlabelled vertices in G_i;
T	denotes the set of the representatives of the strongly connected components of the trap. (Note that elements of F form trivial components of the trap.);
$V(r)$	denotes the vertex $p' \notin K(r)$ to which an edge of G_i leads from $K(r)$, provided that such an edge exists. (For uniqueness *see* Lemma 2.3.)

Every strongly connected component K of the trap has to be examined for transitivity of the relation ε_K on Q defined as follows: $(q_1, q_2) \in \varepsilon_K$ iff $\{q_1, q_2\} \in K$. We shall construct the smallest equivalence ε containing ε_K and we shall examine (during the construction of the equivalence) whether $\varepsilon = \varepsilon_K$. If $\varepsilon = \varepsilon_K$, then ε_K is transitive and there is no edge of type T leading from K. If $(q_1, q_2) \in \varepsilon \setminus \varepsilon_K$, then there exists an edge (r, p) of type T, where r is a representative of K and $p = \{q_1, q_2\}$. (*See* definition of $G(M)$.)

Let us describe the construction of ε. We shall start sith $\varepsilon = \Delta_Q$. Let KK be a copy of K. Let us take $p = \{q_1, q_2\} \in KK$. If $(q_1, q_2) \notin \varepsilon$, we have to make a union of $[q_1]$ and $[q_2]$. Before this, we have to check for every $q'_1 \in [q_1]$ and $q'_2 \in [q_2]$ whether $\{q'_1, q'_2\} \in KK$. If so, we shall exclude $\{q'_1, q'_2\}$ from KK; if not the relation ε_K was not transitive, hence we have found an edge of type T and the further construction of ε is not necessary for the time being.

If $\{q'_1, q'_2\} \in KK$ holds for every $q'_1 \in [q_1]$, $q'_2 \in [q_2]$ we shall make a union of $[q_1]$, $[q_2]$ and, if $KK \neq \emptyset$, we shall take another $p \in KK$ and proceed in the same way.

For the purpose of the test of transitivity we shall use the following data structures to store the information concerning the strongly connected component with the representative r:

$E(r, q)$	is the name of the class containing q of the equivalence ε associated with r;
$B(r, n)$	is the class of ε denoted by the name n;
$KK(r)$	is the subset of $K(r)$ as described above;

$M_1(r)$ $M_2(r)$, $M'_1(r)$, $M'_2(r)$ are used to store the states q_1, q_2, q'_1, q'_2 for which $\{q'_1, q'_2\} \notin KK(r)$. These are useful in case the transitivity has to be examined once more for the same representative;

$S(r, -)$ is a permutation of Q such that every cycle of it forms a class $B(r, n)$ for some n.

All sets and equivalence classes in our algorithm are maintained as doubly-linked lists (by forward and backward cyclic mappings), so that we can use one of these mappings for $S(r, -)$. So in our algorithm the maintenance of these cyclic mappings is not explicitely described.

Before starting the algorithm let us initialize:

$$R(p)=p, \quad K(r)=KK(r)=D(r)=\{r\}, \quad H(p)=X,$$

$$L(p)=0, \quad l=0, \quad F=P, \quad T=P, \quad E(r, q)=q,$$

$$B(r, q)=\{q\}, \quad M_1(r)=M_2(r)=M'_1(r)=M'_2(r)=0.$$

```
NEW LABEL: begin; if $F=\emptyset$ then
                begin; if card $T>1$ then;
                          write "reducible";
                          else write "irreducible";
                          stop;

                end;
                else begin; take $r \in F$;
                          $F=F \setminus \{r\}$;
                          $l=l+1$;
                          $L(r)=l$;
                end;
           end NEW LABEL;

LETTER: (finding an edge of type $L$):
        while $D(r) \neq \emptyset$ do
           take $p=\{q_1, q_2\} \in D(r)$;
           if $H(p) \neq \emptyset$ then
L1:          begin; take $x \in H(p)$;
                $H(p)=H(p) \setminus \{x\}$;
                $q'_1=\delta(q_1, x)$; $q'_2=\delta(q_2, x)$;
                if $q'_1 \neq q'_2$ then
                  begin; $p'=\{q'_1, q'_2\}$;
                         if $R(p') \neq r$ then
                           go to EDGE;
                         else;
                  end;
                else;
             end L1;
           else $D(r)=D(r) \setminus \{p\}$;
        end LETTER;
```

```
TRANSITIVITY: (test of transitivity and finding an edge of type $T$):
        while $KK(r) \neq \emptyset$ do
T1: (preparing $q_1, q'_1, q_2, q'_2$ for the while loop T2):
              if $M_1(r) = M'_1(r)$ and $M_2(r) = M'_2(r)$ then
                begin; take $p = \{q_1, q_2\} \in KK(r)$;
                       $KK(r) = KK(r) \setminus \{p\}$;
                       $q'_2 = S(r, q_2)$;
                       if $q'_2 = q_2$ then $q'_1 = S(r, q_1)$;
                       else $q'_1 = q_1$;
                end;
              else begin; $q_1 = M_1(r)$; $q_2 = M_2(r)$;
                          $q'_1 = M'_1(r)$; $q'_2 = M'_2(r)$;
                end;
T2: (examining whether $\{q'_1, q'_2\} \in KK(r)$ for $q'_1 \in [q_1]$, $q'_2 \in [q_2]$):
              while $q'_1 \neq q_1$ or $q'_2 \neq q_2$ do
                    $p' = \{q'_1, q'_2\}$;
                    if $p' \in KK(r)$ then
                      begin; $KK(r) = KK(r) \setminus \{p'\}$;
                             $q'_2 = S(r, q'_2)$;
                             if $q'_2 = q_2$ then $q'_1 = S(r, q'_1)$;
                             else;
                      end;
                    else (interruption due to an edge of type T):
                      begin; $M_1(r) = q_1$; $M_2(r) = q_2$;
                             $M'_1(r) = q'_1$; $M'_2(r) = q'_2$;
                             go to EDGE;
                      end;
              end T2;
T3: (union of classes $[q_1]$, $[q_2]$)
              begin; $n_1 = E(r, q_1)$; $n_2 = E(r, q_2)$;
                     if card $B(r, n_1) <$ card $B(r, n_2)$ then
                       begin; $n = n_2$; $n_0 = n_1$; end;
                     else begin; $n = n_1$; $n_0 = n_2$; end;
                     for each $q \in B(r, n_0)$ do
                         $E(r, q) = n$;
                     end;
                     $B(r, n) = B(r, n) \cup B(r, n_0)$;
                     $M_1(r) = M'_1(r) = q_1$; $M_2(r) = M'_2(r) = q_2$;
              end T3;
            end TRANSITIVITY;
            go to NEW LABEL;

EDGE: (changes due to discovering of a new edge leading from the component $K(r)$ to the vertex $p'$):
        begin; $V(r) = p'$;
               if $L(p') \neq L(r)$ then
E1: (no new cycle has been closed):
              begin; $T = T \setminus \{r\}$;
                     if $L(p') = 0$ then
                       begin; $L(p') = L(r)$;
                              $F = F \setminus \{p'\}$;
                              $r = p'$;
                       end;
                     else go to NEW LABEL;
              end E1;
            else
E2: (a new cycle has been closed):
              begin; $r' = r$; $r_1 = R(p')$;
```

```
E3: (finding the biggest class in the new cycle):
                        while $r_1 \neq r$ do
                            if card $K(r_1) >$ card $K(r')$
                              then $r' = r_1$;
                            else;
                            $r_1 = R(V(r_1))$;
                        end E3;
                        $r = R(V(r'))$;
E4: (creating the new component):
                        while $r \neq r'$ do
                            $T = T \setminus \{r\}$;
                            $K(r') = K(r') \cup K(r)$;
                            $KK(r') = KK(r') \cup K(r)$;
                            for each $p \in K(r)$ do
                                $R(p) = r'$;
                            end;
                            $r = R(V(r))$;
                        end E4;
                   end E2;
              end EDGE;
              go to LETTER;
```

The proof of correctness of algorithm $\mathscr{B}$ follows from Lemmas 2.2-2.8. It remains to prove the time bound.

Statement NEW LABEL takes a fixed amount of time and it is executed at most once for every $p \in P$.

The statements inside the loop LETTER take a fixed amount of time and they are executed at most once for every $p \in P$ and $x \in X$.

Statement EDGE consists of two branches: E1 and E2. Statement E1 takes a fixed amount of time and the number of executions of it is proportional to card P (*see* Lemma 2.4). Statement E2 needs the time proportional to $k.\log k$, where $k = \text{card } P$ (*see* Lemma 2.5).

It remains to prove the time bound of Statement TRANSITIVITY. Statement T1 takes a fixed amount of time and the number of executions of T1 is the same as the number of executions of T2 and T3. In Statement T2 one element is removed from $KK(r)$ for every pair q'_1, q'_2. Thus the time needed for one execution of T2 is proportional to the number of pairs removed from $KK(r)$. The time needed for one execution of Statement T3 is proportional to the cardinality of the smaller of classes $[q_1]$, $[q_2]$. But this cardinality is less or equal to the number of pairs removed from $KK(r)$ for $[q_1]$ and $[q_2]$.

First, assume that no edge of type T was discovered. Then $KK(r) = \emptyset$ after execution of TRANSITIVITY and this strongly connected component is not going to be examined for transitivity any more. Hence the time needed for one execution of TRANSITIVITY is proportional to card $KK(r) \leq$ card $K(r)$.

Now, assume that an edge of type T was discovered. Then the execution of TRANSITIVITY is interrupted and this strongly connected component either is not examined for transitivity any more or it is examined. In the latter case, a cycle was closed between the old and the new execution of TRANSITIVITY. Our strongly connected component was the largest one in the cycle. Thus the time needed for the new execution is proportional to the time needed for union of strongly connected components contained in the cycle (i. e. proportional to the number of pairs added to old $KK(r)$) plus the time saved in the previous execution due to the interruption when an edge of type T was found.

Therefore, using this trick and Lemma 2.5, the total time needed for Statement TRANSITIVITY is proportional to $k.\log k$, where $k = \text{card } P$. Since card $P \leqq n^2$, where $n = \text{card } Q$, we get that the total time needed for the work of the whole algorithm $\mathscr{B}$ is proportional to $n^2.(m + \log n)$, where $m = \text{card } X$.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT and J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.
2. M. A. ARBIB, *Theories of Abstract Automata*, Englewood Cliffs, N. J., Prentice-Hall Inc., 1969.
3. G. BIRKHOFF, *Lattice Theory*, A.M.S., 1968.
4. G. BIRKHOFF, *Subdirect Unions in Universal Algebra*, Bull. Amer. Math. Soc., 50, 1944, pp. 764-768.
5. A. H. CLIFFORD and G. B. PRESTON, *The Algebraic Theory of Semigroups*, Amer. Math. Soc., 1964.
6. M. DEMLOVÁ, J. DEMEL and V. KOUBEK, *Several Algorithms for Finite Algebras*, F.C.T., 1979, pp. 99-104.
7. M. DEMLOVÁ, J. DEMEL and V. KOUBEK, *Algorithms Deciding Subdirect Irreducibility of Algebras*, to appear.
8. J. HARTMANIS and R. E. STEARNS, *Algebraic Structure Theory of Sequential Machines*, Englewood Cliffs, N. J., Prentice-Hall Inc., 1966.
9. B. M. SCHEIN, *Embedding of Semigroups in Generalized Groups* (russian), Matem. sb., 55, 1961, pp. 397-400.
10. B. M. SCHEIN, *About Transitive Representations of Semigroups* (russian), Uspechi matem. nauk, 18, 1963, pp. 215-222.
11. G. THIERRIN, *Irreducible Automata*, Proc. 25th Summer Meeting of Canadian Math. Congress, 1971, pp. 245-262.
12. E. J. TULLY, *Representation of a Semigroup by Transformations Acting Transitively on a Set*, Amer. J. Math., 83, 1961, pp. 533-541.