Publ. Math. Debrecen
Supplement **60** (2002), 661-677

Trapped automata

By ZARKO POPOVIĆ (Niš), STOJAN BOGDANOVIĆ (Niš),
TATJANA PETKOVIĆ (Niš, Turku) and MIROSLAV ĆIRIĆ (Niš)

Abstract. Certain generalized varieties of automata and the pseudovarieties consisting of their finite members, were introduced by PETKOVIĆ, ĆIRIĆ and BOGDANOVIĆ in [16], where the structural properties of automata from these generalized varieties and of their transition semigroups were studied. The main purpose of this paper is to give algorithms for testing the membership of a finite automaton to the pseudovarieties of finite trapped, locally trap-directable and trap-directable automata, and also, to determine, for each of these pseudovarieties, the least congruence on a finite automaton whose related factor automaton belongs to it. The related problems concerning finite directable automata were considered by IMREH and STEINBY in [12], whereas DEMEL, DEMLOVÁ and KOUBEK in [8] treated similar problems on algebras.

1. Introduction and preliminaries

Automata considered throughout this paper will be automata without outputs in the sense of the definition from the book by GÉCSEG and PEÁK [9]. It is well known that automata without outputs, with the input alphabet X, can be considered as unary algebras of type indexed by X, so the notions such as a *congruence*, *homomorphism*, *generating set* etc., will have their usual algebraic meanings (see, for example, [4]). In order to simplify notation, an automaton with the state set A will be also denoted by the same letter A. For any considered automaton A, its input alphabet will be denoted by X, and the free monoid over X, the input monoid of A, is denoted by X^*. Under action of an input word $u \in X^*$, the automaton A goes from a state a into the state that will be denoted

Mathematics Subject Classification: 68Q45, 68Q70.
Key words and phrases: Pseudovarieties of automata, trapped automata, trap-directable automata.

by au. For $k \in \mathbb{N}^0$, where $\mathbb{N}^0$ denotes the set of all non-negative integers, we write $X^{\leq k} = \{u \in X^* \mid |u| \leq k\}$, where $|u|$ is the length of the word u.

For a subset H of A, by $S(H)$ we denote the *subautomaton* of A *generated by* H, and by $D(H)$ we denote the *dual subautomaton* of A *generated by* H. In other words, $S(H) = \{b \in A \mid (\exists a \in H)(\exists u \in X^*)\ au = b\}$, and $D(H) = \{b \in A \mid (\exists a \in H)(\exists u \in X^*)\ bu = a\}$. Especially, for $a \in A$, the *monogenic subautomaton* and the *monogenic dual subautomaton* generated by a are denoted by $S(a)$ and $D(a)$, respectively.

A state $a \in A$ is called a *trap* of A if $au = a$ for every word $u \in X^*$. The set of all traps of A is denoted by $Tr(A)$. A state $a \in A$ is *reversible* if for every word $u \in X^*$ there exists a word $v \in X^*$ such that $auv = a$. The set of all reversible states of A, called the *reversible part* of A, is denoted by $R(A)$. An automaton A is *reversible* if every its state is reversible. If for every $a, b \in A$ there exists $u \in X^*$ such that $b = au$, then the automaton A is called *strongly connected*. On the other hand, A is called *connected* if for every $a, b \in A$ there exists $u, v \in X^*$ such that $au = bv$, whereas A is said to be *trap-connected* if it is connected and has a trap, or equivalently, if it has a trap a_0 and for every $a \in A$ there exists a word $u \in X^*$ such that $au = a_0$. Two states $a, b \in A$ are *mergeable* if there exists a word $u \in X^*$ such that $au = bu$.

Let $u \in X^*$. An automaton A is called *u-trapped* if $au \in Tr(A)$ for every $a \in A$, and in this case u is called a *trapping word* of A. If $au = bu$ for every $a, b \in A$, then A is called *u-directable*, u is called a *directing word* of A and the set of all directing words of A is denoted by $DW(A)$. If A is u-directable and has a trap, or equivalently, if it is u-trapped and has a unique trap, then it is called *u-trap-directable*. An automaton A is called *trapped* (resp. *directable*, *trap-directable*) if there exists a word $u \in X^*$ such that A is *u-trapped* (resp. *u-directable*, *u-trap-directable*). A word $u \in X^*$ is called a *locally trap-directing word* of an automaton A if every monogenic subautomaton of A is u-trap-directable, i.e., if $apuq = au$ holds for every $a \in A$ and every $p, q \in X^*$. An automaton having a locally trap-directable word is called a *locally trap-directable automaton*. Trapped, trap-directable and locally trap-directable automata were introduced in [16], whereas directable automata were first studied in [5]. More information about these classes of automata can be find in the survey paper [2].

For a subautomaton B of an automaton A the *Rees congruence* ϱ_B is defined by: $(a, b) \in \varrho_B \Leftrightarrow a = b$ or $a, b \in B$. The factor automaton A/ϱ_B is denoted by A/B and the automaton A is said to be an *extension* of B by an automaton C (with a trap), where $A/B \cong C$.

An automaton A is the *direct sum* of its subautomata A_α, $\alpha \in Y$, in notation $A = \sum_{\alpha\in Y} A_\alpha$, if $A = \bigcup_{\alpha\in Y} A_\alpha$ and $A_\alpha \cap = A_\beta = \varnothing$, for every $\alpha, \beta \in Y$ such that $\alpha \neq \beta$. Automata A_α, $\alpha \in Y$, are *direct summands* of A, and they determine a congruence on A called a *direct sum congruence* and the corresponding decomposition is called a *direct sum decomposition*. By the *greatest direct sum decomposition* of A we mean the decomposition which corresponds to the least direct sum congruence on A. More on direct sum decompositions can be find in [7].

In the algorithms that follow here we consider finite automata, i.e. automata with finite state sets and input sets. The set state of cardinality n is denoted by $\{1, 2, \ldots, n\}$. The automaton A is given by its transition table $T = (T[i, x])_{i\in A, x\in X}$, where $T[i, x] = j$ if $ix = j$, for $i, j \in A$ and $x \in X$. A *list* is defined as a linearly ordered sets of data. For a list L, $i \to L$ means that an element i is put on L, whereas $i \leftarrow L$ means that the first element i of L is deleted from L. The empty list is denoted by $\varnothing$.

For undefined notions and notation we refer to [9] and [4].

As was proved by Starke in [17], a finite automaton is directable if and only if any two its states are mergeable. As an immediate consequence of this statement we obtain the following result which will be very useful in our further considerations.

Lemma 1. *A finite automaton is trap-connected if and only if it is trap-directable.*

2. Tests for trappedness, trap-directability, and local trap-directability

As a tool which can save operating time of an algorithm which tests a finite automaton for directability, Imreh and Steinby used in [12] the notion of an inverse transition table of an automaton. Here we introduce and use a related notion – the inverse vector of an automaton.

In the algorithms that follow we shall assume that A is a finite automaton with m input letters and n states, which are denoted by $1, 2, \ldots, n$, i.e. $A = \{1, 2, \ldots, n\}$. If the automaton A is given by its transition table $T = (T[i, x])_{i\in A,\, x\in X}$, then the *inverse vector* $I = (I[i])_{i\in A}$ of A is formed in the following way: for every state $i \in A$, the set $I[i]$ consists of all states $j \in A$ for which there exists an input letter $x \in X$ which leads from the state j into the state i, i.e.

$$I[i] = \{j \in A \mid (\exists x \in X)\, T[j, x] = i\}.$$

Any $j \in I[i]$ is called an *immediate predecessor* of the state i, whereas i is called an *immediate successor* of j.

We give the following simple algorithm which generates the inverse vector of a given finite automaton.

Algorithm: The inverse vector
Input: the set A of states of an automaton A,
the set X of input letters of A,
the transition table T of A.
Output: The inverse vector $I = (I[i])_{i \in A}$.
Procedure:
Step 1. Initialization:
for $i \in A$ **do** $I[i] := \varnothing$.
Step 2. **for** $i \in A$ **do**
for $x \in X$ **do**
$i \to I[T[i, x]]$.

It is easy to check that this algorithm operates in time $\mathcal{O}(mn)$.

Let us pass to the trappedness problem. The minimal length of trapping words of a trapped automaton can be estimated using the estimation for the minimal length of directing words of directed automata. For a directable automaton A the number $d(A)$ is defined by $d(A) = \min\{|u| \mid u \in DW(A)\}$, and for $n \in \mathbb{N}$, the number $d(n)$ is defined as

$$d(n) = \max\{d(A) \mid A \text{ is a directable automaton with } |A| = n\}.$$

The well-known Černý's conjecture says that $d(n) \leq (n-1)^2$. In the case of trapped automata we have the following estimation:

Lemma 2. *Let A be a trapped automaton with n states and t traps. Then the minimal length of trapping words of A is less or equal than $d(n-t+1)$.*

The proof of this lemma follows immediately by Theorem 3 of [16] which characterizes trapped automata as extensions of discrete automata by trap-directable automata. If we assume that Černý's conjecture holds, which is the best we can hope for, then for an automaton with n states and t traps we have to check the existence of a trapping word in the set $X^{\leq (n-t)^2}$. But, this algorithm is not so fast, and in the sequel we are looking for some better algorithm for testing trappedness.

Let a sequence $\{P_k\}_{k \in \mathbb{N}}$ of subsets and a subset P of an automaton A

be defined as follows:

$$P_k = \{a \in A \mid (\exists u \in X^{\leq k})\ au \in Tr(A)\}, \quad \text{for} \quad k \in \mathbb{N},$$
$$P = \bigcup_{k\in\mathbb{N}} P_k = \{a \in A \mid (\exists u \in X^*)\ au \in Tr(A)\}.$$

We prove the following

Theorem 1. *Let A be a finite automaton with n states and a nonempty set of traps. Then the following conditions hold:*
(a) *$\{P_k\}_{k\in\mathbb{N}}$ is an increasing sequence of sets;*
(b) *the sets P_k can be computed as follows:*
  (1) *$P_0 = Tr(A)$,*
  (2) *$P_{k+1} = P_k \cup \{a \in A \mid (\exists x \in X)\ ax \in P_k\}$, for every $k \in \mathbb{N}$;*
(c) *if $P_k = P_{k+1}$, for some $k \in \mathbb{N}$, then $P_k = P_{k+m} = P$, for every $m \in \mathbb{N}$;*
(d) *there exists $k \in [0, n-1]$ such that*

$$Tr(A) = P_0 \subseteq P_1 \subseteq \ldots \subseteq P_k = P_{k+1} = \ldots = P;$$

(e) *$P = D(Tr(A))$;*
(f) *A is trapped if and only if $A = P$.*

Proof. (a) This claim is obvious.

(b) It is clear that $P_0 = Tr(A)$. We shall prove the second equality. By (a) we have that $P_k \subseteq P_{k+1}$. Consider an arbitrary $a \in P_{k+1} \setminus P_k$. Since $a \in P_{k+1}$, then there exists a word $u \in X^{\leq k+1}$ such that $au \in Tr(A)$. Let $u = xu'$ for some $x \in X$ and $u' \in X^{\leq k}$. Then $(ax)u' = au \in Tr(A)$, what implies $ax \in P_k$. Therefore, $P_{k+1} \subseteq P_k \cup \{a \in A \mid (\exists x \in X)\ ax \in P_k\}$.

To prove the opposite inclusion, consider $a \in A$ such that $ax \in P_k$ for some $x \in X$. Then there exists a word $u \in X^{\leq k}$ such that $(ax)u \in Tr(A)$, and for a word $v = xu \in X^{\leq k+1}$ we have that $av \in Tr(A)$, so $a \in P_{k+1}$.

(c) This follows immediately by the equality (2) of (b).

(d) This statement is an immediate consequence of (a), (c) and the fact that the automaton A has n states.

(e) If $a \in P$, then there exists $k \in \mathbb{N}$ such that $a \in P_k$, i.e. $au \in Tr(A)$, for some word $u \in X^{\leq k}$, and hence, $a \in D(Tr(A))$. On the other hand, if $a \in D(Tr(A))$, then there exists $u \in X^*$ such that $au \in Tr(A)$, what yields $a \in P_k \subseteq P$, where $k = |u|$.

(f) Let A be a trapped automaton, let u be a trapping word of A and let $|u| = k$. Then $A \subseteq P_k \subseteq P$, so we have proved that $A = P$. Conversely, if $A = P$, then for any $a \in A$ there exists $u \in X^*$ such that $au \in Tr(A)$, so the Rees factor automaton $A/Tr(A)$ is trap-connected, and according to Lemma 1, $A/Tr(A)$ is trap-directable. Therefore, A is an extension of

a discrete automaton by a trap-directable automaton, and by Theorem 3 of [16], we have that A is a trapped automaton. □

Using the above theorem we give an algorithm which tests finite automata for trappedness. Namely, we form an increasing sequence of sets, where the least one is $Tr(A)$ and the largest one is P, which is the union of all P_k, $k \in \mathbb{N}^0$. The set P is the dual subautomaton generated by $Tr(A)$ and by (f) of Theorem 1 we have that automaton A is trapped if and only if $A = P$.

Algorithm: Test for trappedness

Input: The set A of states of an automaton A;
The set X of input letters of A;
The transition table T of A.

Output: YES, if A is trapped, or NO, if A is not trapped.

Auxiliary data structures::
: The list L;
: The Boolean vector $V = (V[i])_{i \in I}$;
: The Boolean variable t.

Procedure:

Step 1. Initialization:
for $i \in A$ **do** $V[i] := 0$;
$L := \varnothing$.

Step 2. Formation of the inverse vector $I = (I[i])_{i \in A}$.

Step 3. Formation of the list of traps:
for $i \in A$ **do**
$t := 0$;
for $x \in X$ **do**
if $T[i, x] \neq i$ **then** $t := 1$;
if $t = 0$ **then** $i \to L$, $V[i] := 1$.

Step 4. **while** $L \neq \varnothing$ **do**
$i \leftarrow L$;
for $j \in I[i]$ **do**
if $V[j] = 0$ **then** $j \to L$, $V[j] := 1$.

Step 5. **for** $i \in A$ **do**
if $V[i] = 0$ **then** STOP and NO;
YES.

Let us describe how the algorithm works. The main role of the vector V is to indicate whether a state i belongs to the set P ($V[i] = 1$) or not ($V[i] = 0$). The algorithm starts with the empty list L, and in Step 3 all traps are included on L and registered as members of P. Further, if a state i has been registered as a member of P and put on L, then it is kept on L until it is replaced on L by all its immediate predecessors which have not been yet on the list, when these predecessors are also registered to belong to P. Since we simultaneously set $i \to L$ and $V[i] := 1$, then

we have that i has not been yet on L if and only if $V[i] = 0$. The Step 4 finishes when the list L is emptied, and then we have that $i \in P$ if and only if $V[i] = 1$. Therefore, by Theorem 1 it follows that A is trapped if and only if $V[i] = 1$ for every $i \in A$. It can be verified that the algorithm operates in time $\mathcal{O}(mn)$.

By a slight modification of the previous algorithm we obtain an algorithm which tests finite automata for trap-directability. Namely, after we form the set of traps, it should be checked whether this set contains only one element. If so, then we proceed by testing for trappedness, otherwise we immediately conclude that the automaton A is not trap-directable.

Concerning an algorithm which tests finite automata for local trap-directability, its theoretical base is given by the following theorem.

Theorem 2. *A finite automaton A is locally trap-directable if and only if the following conditions are satisfied:*
(a) $Tr(A) \neq \varnothing$;
(b) $D(a) \cap D(b) = \varnothing$, *for all* $a, b \in Tr(A)$ *such that* $a \neq b$;
(c) $A = \bigcup\{D(a) \mid a \in Tr(A)\}$.

Proof. Let A be locally trap-directable. It is clear that (a) holds. Consider $a, b \in Tr(A)$ such that $a \neq b$. If $c \in D(a) \cap D(b)$, then $a, b \in S(c)$, which is impossible because by the hypothesis it follows that $S(c)$ is a trap-directable automaton and it can not have two different traps. Therefore, we conclude that $D(a) \cap D(b) = \varnothing$.

To prove (c), consider an arbitrary state $b \in A$. By the hypothesis, the monogenic subautomaton $S(b)$ of A is trap-directable, and if a is the unique trap of $S(b)$, then $b \in D(a)$. Thus, (c) holds.

Conversely, let (a), (b) and (c) hold and consider an arbitrary state $c \in A$. By (c) it follows that $c \in D(a)$ for some $a \in Tr(A)$, and if $cu \in D(b)$ for some $u \in X^*$ and $b \in Tr(A)$, $b \neq a$, then we have that $c \in D(cu) \subseteq D(b)$, which contradicts the statement (b). Thus, we conclude that $cu \in D(a)$, for each $u \in X^*$, which means that $S(c) \subseteq D(a)$. Now it is clear that $S(c)$ is a trap-connected automaton with the trap a, and by Lemma 1, $S(c)$ is trap-directable. Hence, A is a locally trap-directable automaton. □

In other words, the previous theorem says that an automaton A is locally trap-directable if and only if every dual subautomaton of A generated by a trap is a subautomaton of A and A is the direct sum of these subautomata. According to this result we can give the following algorithm which tests local trap-directability.

Algorithm: Test for local trap-directability

Input: The set A of states of an automaton A;
The set X of input letters of A;
The transition table T of A.
Output: YES, if A is locally trap-directable, or NO, if A is not locally trap-directable.
Auxiliary data structures::
: Lists Tr and L;
: Boolean vectors $V = (V[i])_{i \in I}$ and $U = (U[i])_{i \in I}$.
Procedure:
Step 1. Initialization:
for $i \in A$ **do** $V[i] := 0$;
$Tr := \varnothing$, $L := \varnothing$.
Step 2. Formation of the inverse vector $I = (I[i])_{i \in A}$.
Step 3. Formation of the list of traps:
for $i \in A$ **do**
for $x \in X$ **do**
if $T[i, x] = i$ **then** $i \to Tr$.
Step 4. **while** $Tr \neq \varnothing$ **do**
$i \leftarrow Tr$, $i \to L$, $V[i] := 1$;
for $j \in A$ **do** $U[j] := 0$;
Step 4.1. **while** $L \neq \varnothing$ **do**
$j \leftarrow L$;
for $l \in I[j]$ **do**
if $U[l] = 0$ **then**
if $V[l] = 1$ **then** STOP and NO
else $l \to L$, $U[l] := 1$, $V[l] := 1$.
Step 5. **for** $i \in A$ **do**
if $V[i] = 0$ **then** STOP and NO;
YES.

In contrast to the test for trappedness, in which all dual subautomata generated by traps are built simultaneously, here we build any of them separately and we check whether they are mutually disjoint. Namely, when we form the list Tr of all traps (Step 3), we do not include the whole list Tr on the list L, but only one trap. Another trap is not put on L until L is emptied. Step 4 has to register the states which belong to the set $P = \bigcup\{D(i) \mid i \in Tr(A)\}$, by means of the vector V, whereas for a trap i, the role of Step 4.1 is to register the members of $D(i)$, using the vector U which we reset any time before we repeat Step 4.1. In Step 4.1 we also use V to check whether $D(i)$ is disjoint with the previously formed principal dual subautomata. If it is not disjoint with some of them, then the algorithm is stopped immediately and the answer is NOT. The answer is YES if and only if the algorithm is finished regularly and at the end we have that $V[i] = 1$ for every $i \in A$.

3. The least trapping, trap-directing and locally trap-directing congruence

A congruence relation θ on an automaton A is called *trapping*, *trap-directing* or *locally trap-directing* if the related factor automaton A/θ is trapped, trap-directable or locally trap-directable, respectively. In this section we shall construct the least trapping, trap-directing and locally trap-directing congruence on an arbitrary finite automaton A.

By Theorem 3 of [14], every finite automaton can be uniquely represented as an extension of a reversible automaton by a trap-directable automaton. On the other hand, by the well-known result given by THIERRIN in [18] and GLUSHKOV in [11], it follows that a reversible automaton can be uniquely represented as the direct sum of its strongly connected subautomata. These representations will be very useful in the next theorem.

Theorem 3. *Let a finite automaton A be represented as an extension of a reversible automaton B by a trap-directable automaton C, and let B be represented as the direct sum of its strongly connected subautomata B_α, $\alpha \in Y$. Then the relation τ on A defined by*

$$(a,b) \in \tau \quad \Leftrightarrow \quad a = b \ \text{ or } \ a, b \in B_\alpha, \ \text{for some } \alpha \in Y,$$

is the least trapping congruence on A.

PROOF. Let σ be the least direct sum congruence on B, i.e. the congruence whose classes are B_α, $\alpha \in Y$. Then τ can be written as $\tau = \sigma \cup \Delta_A$, where Δ_A denotes the equality relation on A, and by Theorem 4.1 of [1] it follows that τ is a congruence relation on A.

Let u be a directing word of $C \cong A/B$. Consider arbitrary $a \in A$ and $v \in X^*$. Then $au \in B$, that is $au \in B_\alpha$, for some $\alpha \in Y$, whence it follows that $auv \in B_\alpha$, so $(auv, au) \in \tau$. This means that A/τ is a trapped automaton, i.e. τ is a trapping congruence on A.

Let θ be an arbitrary trapping congruence on A. Then there exists $v \in X^*$ such that $(avw, av) \in \theta$, for all $a \in A$ and $w \in X^*$. Consider $(a,b) \in \tau$ such that $a \neq b$. Then $a, b \in B_\alpha$, for some $\alpha \in Y$, and since B_α is strongly connected, then $avp = a$ and $avq = b$, for some words $p, q \in X^*$. Now we have that

$$a = avp, \quad (avp, av) \in \theta, \quad (av, avq) \in \theta \quad \text{and} \quad avq = b,$$

whence $(a,b) \in \theta$. Therefore, we have proved that $\tau \subseteq \theta$, so τ is the least trapping congruence on A. □

According to the previous theorem, in order to construct the least trapping congruence on some finite automaton, i.e. to determine its classes, we have to find all its strongly connected subautomata. Each of them is one non-trivial class and the elements that are not included in any of these subautomata represent one-element classes. Before giving an algorithm for determining the strongly connected subautomata of a finite automaton we have one auxiliary result.

Lemma 3. *Let A be an automaton and let a be its arbitrary state. Then:*
(a) *$a \in R(A)$ if and only if $S(a) \subseteq D(a)$;*
(b) *if $a \in R(A)$ then $S(a) \subseteq R(A)$ and $(D(a) \setminus S(a)) \cap R(A) = \varnothing$.*

Proof. (a) This assertion was proved in Lemma 1 of [14].

(b) Let $a \in R(A)$. By Lemma 2 of [14], $R(A)$ is a subautomaton of A, whence $S(a) \subseteq R(A)$. Suppose now that $b \in (D(a) \setminus S(a)) \cap R(A)$. By $b \in D(a)$ we have that $bu = a$, for some $u \in X^*$, whereas $b \in R(A)$ implies that there exists $v \in X^*$ such that $buv = b$. But then $b = buv = av$, i.e. $b \in S(a)$, so we have obtained a contradiction. Thus, $(D(a) \setminus S(a)) \cap R(A) = \varnothing$. □

Now we can expose the algorithm for determining the strongly connected components of a finite automaton. Note that an algorithm for finding the strongly connected components of a graph can not be used here because the notions of strong connectivity in Graph Theory and Automata Theory do not coincide.

Algorithm: Strongly connected subautomata
Input: The set A of states of an automaton A;
The set X of input letters of A;
The transition table T of A.
Output: The lists $R_1, \ldots, R_k$ which represent all strongly connected subautomata of A.
Auxiliary data structures::
: Lists C, S and D;
: Boolean vectors $s = (s[i])_{i \in I}$ and $d = (d[i])_{i \in I}$,
: The number k.
Procedure:
Step 1. Initialization:
$C := \varnothing$, $k := 0$.
Step 2. Formation of the inverse vector $I = (I[i])_{i \in A}$.
Step 3. **for** $i \in A$ **do** $i \to C$.

Step 4. **while** $C \neq \varnothing$ **do**
Step 4.1. **for** $j \in A$ **do** $s[j] := 0$, $d[j] := 0$;
$S := \varnothing$, $D := \varnothing$,
$i \leftarrow C$, $i \to S$, $i \to D$;
$s[i] := 1$, $d[i] := 1$;
Step 4.2. **while** $D \neq \varnothing$ **do**
$j \leftarrow D$;
for $l \in I[j]$ **do**
if $d[l] = 0$ **then** $l \to D$, $d[l] := 1$;
Step 4.3. **while** $S \neq \varnothing$ **do**
$j \leftarrow S$;
for $x \in X$ **do**
if $s[T[j,x]] = 0$ **then**
if $d[T[j,x]] = 0$ **then goto** Step 4.1
else $s[T[j,x]] := 1$, $T[j,x] \to S$;
Step 4.4. $k := k + 1$;
for $j \in A$ **do**
if $s[j] = 1$ **then** $j \to R_k$, $j \leftarrow C$;
else if $d[j] = 1$ **then** $j \leftarrow C$.

This algorithm works as follows. After initialization and formation of the inverse vector, in Step 3 we put all states on the list C. When we start to check whether a state i is reversible, we delete it from C and put it on the lists S and D (Step 4.1). These lists are used to generate $S(i)$ and $D(i)$, and the states that belong to $S(i)$ and $D(i)$ are registered by means of the vectors s and d, which we reset at the beginning of each cycle of Step 4. The dual subautomaton $D(i)$ is generated in Step 4.2, and in Step 4.3 we simultaneously generate the subautomaton $S(i)$ and check whether $S(i) \subseteq D(i)$. If a state that belongs to $S(i) \setminus D(i)$ is found, this means that i is not reversible and we immediately stop the current cycle and start the next cycle of Step 4. Otherwise, if $S(i) \subseteq D(i)$, then i is reversible and $S(i)$ is the strongly connected subautomaton containing it, so we put all states from $S(i)$ on the list R_k and we delete them from C. On the other hand, by Theorem 3 it follows that none of the states from $D(i) \setminus S(i)$ lies in a strongly connected subautomaton of A, so we do not need to check these states and we delete them from the list C.

The next theorem describes the least trap-directing congruence on an automaton.

Theorem 4. *Let a finite automaton A be represented as an extension of a reversible automaton B by a trap-directable automaton C. Then the Rees congruence ϱ_B on A is the least trap-directing congruence on A.*

Proof. It is evident that ϱ_B is a trap-directing congruence on A, so it remains to prove that it is the least trap-directing congruence. Let θ be an arbitrary trap-directing congruence on A. Then there exists a word $u \in X^*$ such that $(auv, bu) \in \theta$, for all $a, b \in A$ and $v \in X^*$. Consider $(a, b) \in \varrho_B$ such that $a \neq b$. Then $a, b \in B$ and since B is a reversible automaton, then $a = aup$ and $b = buq$, for some words $p, q \in X^*$. Now we have that

$$a = aup, \quad (aup, bu) \in \theta, \quad (bu, buq) \in \theta \quad \text{and} \quad buq = b,$$

which implies $(a, b) \in \theta$. Therefore, $\varrho_B \subseteq \theta$, so we have proved that ϱ_B is the least trap-directing congruence on A. □

Corollary 1. *Let A be a finite automaton and let $R(A)$ be its reversible part. Then A is trap-directable if and only if $|R(A)| = 1$.*

It is evident that the algorithm for finding the least trap-directing congruence on a finite automaton A can be obtained by a slight modification of the previous algorithm concerning the least trapping congruence on A. Namely, we do not need to separate the strongly connected components, we just have to collect all reversible states on one list.

Before we prove a theorem which characterizes the least locally trap-directing congruence on an automaton, we give an auxiliary result which can be verified easily.

Lemma 4. *Let $A = \sum_{\alpha \in Y} A_\alpha$ be a direct sum decomposition of an automaton A, for each $\alpha \in Y$ let θ_α be a congruence relation on A_α and let $\theta = \bigcup_{\alpha \in Y} \theta_\alpha$. Then θ is a congruence relation on A.*

Now we are ready to prove the following:

Theorem 5. *Let A be a finite automaton, let $A = \sum_{i=1}^{k} A_i$ be the greatest direct sum decomposition of A, for each $i \in [1, k]$ let ϱ_i be the least trap-directing congruence on A_i, and let $\varrho = \bigcup_{i=1}^{k} \varrho_i$. Then ϱ is the least locally trap-directing congruence on A.*

Proof. By Lemma 4 it follows that ϱ is a congruence relation on A. Next we prove that A/ϱ is a locally trap-directable automaton. For any $i \in [1, k]$ let u_i be an arbitrary trap-directing word of A_i/ϱ_i and let $u = u_1 u_2 \dots u_k$. Then u is a trap-directing word of A_i/ϱ_i, for every $i \in [1, k]$. Let us prove that u is a locally trap-directing word of A/ϱ.

Consider arbitrary $a \in A$ and $p, q \in X^*$. Let $i \in [1, k]$ such that $a \in A_i$. Since A_i/ϱ_i is trap-directable with u as one of its trap-directing words, then we have that $(apuq, apu) \in \varrho_i$ and $(apu, au) \in \varrho_i$, whence $(apuq, au) \in \varrho_i$. Therefore, $(apuq, au) \in \varrho$, which means that A/ϱ is a locally trap-directable automaton with u as one of its locally trap-directing words.

Let θ be an arbitrary congruence relation on A such that A/θ is a locally trap-directable automaton. This means that there exists a word $u \in X^*$ such that $(apuq, au) \in \theta$, or equivalently,

$$(apu, au) \in \theta \quad \text{and} \quad (auq, au) \in \theta, \tag{1}$$

for all $a \in A$ and $p, q \in X^*$. For any $i \in [1, k]$ let θ_i be the restriction of θ onto A_i, i.e. $\theta_i = \theta \cap (A_i \times A_i)$. Then θ_i is a congruence relation on A_i and we shall prove that it is a trap-directing congruence on A_i.

Consider $a, b \in A_i$ such that $S(a) \cap S(b) \neq \varnothing$, i.e. $av = bw$, for some $v, w \in X^*$. Then by (1) it follows that $(avu, au) \in \theta_i$ and $(bwu, bu) \in \theta_i$, whence $(au, bu) \in \theta_i$. Consider now arbitrary $a, b \in A_i$. By Theorem 3.2 of [7], there exist $c_1, c_2, \ldots, c_j \in A_i$ such that

$$S(a) \cap S(c_1) \neq \varnothing,\ S(c_1) \cap S(c_2) \neq \varnothing,\ \cdots,\ S(c_j) \cap S(b) \neq \varnothing,$$

and by the previously proved fact we obtain that

$$(au, c_1u) \in \theta_i,\ (c_1u, c_2u) \in \theta_i,\ \ldots\ , (c_ju, bu) \in \theta_i,$$

so $(au, bu) \in \theta_i$. On the other hand, (1) yields $(auq, au) \in \theta_i$, for each $q \in X^*$. Therefore, we conclude that $(auq, bu) \in \theta_i$, for each $q \in X^*$, which means that θ_i is a trap-directing congruence on A_i. By the hypothesis, ϱ_i is the least trap-directing congruence on A_i, so $\varrho_i \subseteq \theta_i$. Therefore,

$$\varrho = \bigcup_{i=1}^{k} \varrho_i \subseteq \bigcup_{i=1}^{k} \theta_i = \theta,$$

so we have that ϱ is the least locally trap-directing congruence on A. □

According to the previous theorem, an algorithm for finding the least locally trap-directing congruence on a finite automaton can be divided into two parts. In the first one the considered automaton is decomposed into the greatest direct sum decomposition, and in the second one the reversible parts of all summands of this decomposition are found. Therefore, it remains to give an algorithm for decomposition of a finite automaton into the greatest direct sum decomposition.

As was proved by Ćirić and Bogdanović in [7], for a state a of an automaton A, the summand containing a in the greatest direct sum decomposition of A equals the principal filter $F(a)$ generated by a. It was also proved in the same paper that $F(a)$ can be computed as follows

$$F(a) = \bigcup_{k\in\mathbb{N}^0} U_k(a),$$

where $\{U_k(a)\}_{k\in\mathbb{N}^0}$ is an increasing sequence of sets defined by

$$U_0(a) = \{a\} \quad \text{and} \quad U_{k+1}(a) = D(S(U_k(a))), \text{ for any } k \in \mathbb{N}^0.$$

If A is a finite automaton with n states, then $F(a) = U_k(a)$, for some $k \in [1, n]$, and in this case we can compute every summand in the greatest direct sum decomposition starting from some states and applying alternately the operators $S : H \mapsto S(H)$ and $D : H \mapsto D(H)$ finitely many times. But, here we give another algorithm with synchronous application of the operators S and D. For a subset H of an automaton A, we define the set $A(H)$ of *adjacent states* of H by

$$A(H) = H \cup \{b \in A \mid (\exists a \in H)(\exists x \in X)\, ax = b \text{ or } bx = a\},$$

and we prove the following theorem.

Theorem 6. *Let a be a state of an automaton A, $A_0(a) = \{a\}$ and $A_{k+1}(a) = A(A_k(a))$ for $k \in \mathbb{N}^0$. Then $\{A_k(a)\}_{k\in\mathbb{N}^0}$ is an increasing sequence of sets and $F(a) = \bigcup_{k\in\mathbb{N}^0} A_k(a)$.*

Proof. First we prove that $A_k(a) \subseteq U_k(a)$, for each $k \in \mathbb{N}^0$. This inclusion is evident for $k = 0$. Suppose that $A_k(a) \subseteq U_k(a)$ for some $k \in \mathbb{N}^0$, and consider an arbitrary $c \in A_{k+1}(a)$. There are three possibilities: $c \in A_k(a)$, or $c = bx$, for $b \in A_k(a)$ and $x \in X$, or $cx = b$, for $b \in A_k(a)$ and $x \in X$. In the first case $c \in A_k(a) \subseteq U_k(a) \subseteq U_{k+1}(a)$. In the second case we have that $b \in U_k(a)$, so $c \in S(U_k(a)) \subseteq D(S(U_k(a))) = U_{k+1}(a)$, whereas in the third case it follows that $b \in U_k(a) \subseteq S(U_k(a))$, whence $c \in D(S(U_k(a))) = U_{k+1}(a)$. Hence, by induction we obtain that $A_k(a) \subseteq U_k(a)$, for every $k \in \mathbb{N}^0$, which yields $\bigcup_{k\in\mathbb{N}^0} A_k(a) \subseteq F(a)$.

To prove the opposite inclusion, it is enough to prove that $U_i(a) \subseteq \bigcup_{k\in\mathbb{N}^0} A_k(a)$ for every $i \in \mathbb{N}^0$. It is clear that this is satisfied for $i = 0$. Suppose now that this inclusion holds for some $i \in \mathbb{N}^0$ and consider an arbitrary $d \in U_{i+1}(a)$. Then $du = c$, for some $c \in S(U_i(a))$ and $u \in X^*$, and $c = bv$, for some $b \in U_i(a)$ and $v \in X^*$. Then $u \in X^r$ and $v \in X^s$, for some $r, s \in \mathbb{N}^0$, whereas by the hypothesis it follows that $b \in$

$A_t(a)$, for some $t \in \mathbb{N}^0$, so we have that $d \in A_{r+s+t}(a)$. Thus, $U_{i+1}(a) \subseteq \bigcup_{k\in\mathbb{N}^0} A_k(a)$, and by induction we obtain that $U_i(a) \subseteq \bigcup_{k\in\mathbb{N}^0} A_k(a)$, for every $i \in \mathbb{N}^0$. This finally yields $F(a) \subseteq \bigcup_{k\in\mathbb{N}^0} A_k(a)$, which was to be proved. □

Now we are ready to give the following algorithm for decomposition of a finite automaton into the greatest direct sum decomposition.

Algorithm: The greatest direct sum decomposition:

Input: The set A of states of an automaton A;
The set X of input letters of A;
The transition table T of A.

Output: The lists $F_1, \ldots, F_k$ which represent the summands in the greatest direct sum decomposition of A, i.e. the principal filters of A.

Auxiliary data structures::
: Lists C and L;
: The Boolean vector $V = (V[i])_{i\in I}$;
: The number k.

Procedure:

Step 1. Initialization:
$C := \varnothing$, $L = \varnothing$, $k := 0$;
for $i \in A$ **do** $V[i] := 0$.

Step 2. Formation of the inverse vector $I = (I[i])_{i\in A}$.

Step 3. **for** $i \in A$ **do** $i \to C$.

Step 4. **while** $C \neq \varnothing$ **do**

Step 4.1.
$i \leftarrow C$;
if $V[i] = 0$ **then**
$i \to L$, $k := k + 1$, $i \to F_k$, $V[i] := 1$.

Step 4.2.
while $L \neq \varnothing$ **do**
$j \leftarrow L$;
for $x \in X$ **do**
if $V[T[j, x]] = 0$ **then**
$T[j, x] \to L$, $T[j, x] \to F_k$, $V[T[j, x]] := 1$;
for $l \in I[j]$ **do**
if $V[l] = 0$ **then**
$l \to L$, $l \to F_k$, $V[l] := 1$.

In Step 3, all states are put on the list C. The vector V is used to register the states which have been included on some list F_k. When we choose the first state i from C, we delete it from C and we check whether $V[i] = 0$. If $V[i] = 1$, then this means that it has been put on some F_k, i.e. that $F(i)$ has been already generated. Otherwise, if $V[i] = 0$, then we start to generate $F(i) = F_k$ by putting i on the lists L and F_k, which is immediately

registered by $V[i] := 1$. The members of $F(i)$ are determined in Step 4.2. When we delete a state j from the list L we immediately put on L all its immediate predecessors and successors which have not been yet considered in the current cycle of Step 4.

References

[1] S. Bogdanović, M. Ćirić, T. Petković, B. Imreh and M. Steinby, Traps, cores, extensions and subdirect decompositions of unary algebras, *Fundamenta Informaticae* **34** (1999), 31–40.

[2] S. Bogdanović, B. Imreh, M. Ćirić and T. Petković, Directable automata and their generalizations – A survey, *Proc. VIII Int. Conf. "Algebra and Logic" (Novi Sad, 1998)* (S. Crvenković and I. Dolinka, eds.), *Novi Sad J. Math* **29** (3) (1999), 25–68.

[3] J. A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events, *Proc. Symp. Math. Theory of Automata, Microwave Research Inst. Symp. Ser.* **12** (Brooklyn, 1963), *New York*, 1963, 529–561.

[4] S. Burris and H. P. Sankappanavar, A course in universal algebra, *Springer-Verlag, New York*, 1981.

[5] J. Černý, Poznámka k homogénym experimentom s konecinými automatami, *Mat.-fyz. cas. SAV* **14** (1964), 208–215.

[6] M. Ćirić and S. Bogdanović, Posets of $\mathcal{C}$-congruences, *Algebra Universalis* **36** (1996), 423–424.

[7] M. Ćirić and S. Bogdanović, Lattices of subautomata and direct sum decompositions of automata, *Algebra Colloq.* **6**:1 (1999), 71–88.

[8] J. Demel, M. Demlová and V. Koubek, Fast algorithms constructing minimal subalgebras, congruences, and ideals in a finite algebra, *Theoretical Computer Science* **36** (1985), 203–216.

[9] F. Gécseg and I. Peák, Algebraic Theory of Automata, *Akadémiai Kiadó, Budapest*, 1972.

[10] A. Ginzburg, About some properties of definite, reverse definite and related automata, *IEEE Trans. Electronic Computers* **EC-15** (1966), 809–810.

[11] V. M. Glushkov, Abstraktnaya teoriya avtomatov, *Uspekhi Matem. Nauk* **16** (1961), no. 5, 3–62 [English translation: *The abstract theory of automata*, Russ. Math. Surveys **16** (1961), no. 5, 1–53].

[12] B. Imreh and M. Steinby, Some remarks on directable automata, *Acta Cybernetica* **12** (1995), 23–35.

[13] M. Ito and J. Duske, On cofinal and definite automata, *Acta Cybernetica* **6** (1983), 181–189.

[14] J. Kovačević, M. Ćirić, T. Petković and S. Bogdanović, Decompositions of automata and reversible states, (to appear).

[15] T. Petković, Varieties of automata and semigroups, Ph. D. Thesis, *University of Niš*, 1998, (in Serbian).

[16] T. Petković, M. Ćirić and S. Bogdanović, Decompositions of automata and transition semigroups, *Acta Cybernetica (Szeged)* **13** (1998), 385–403.

[17] P. H. Starke, Abstrakte Automaten, *VEB Deutsher Verlag der Wissenschaften, Berlin*, 1969.

[18] G. Thierrin, Decompositions of locally transitive semiautomata, *Utilitas Mathematica* **2** (1972), 25–32.

ZARKO POPOVIĆ, STOJAN BOGDANOVIĆ
UNIVERSITY OF NIŠ, FACULTY OF ECONOMICS
TRG KRALJA ALEKSANDRA 11, P.O. BOX 121
18000 NIŠ
YUGOSLAVIA

E-mail: zpopovic@orion.eknfak.ni.ac.yu, sbogdan@pmf.pmf.ni.ac.yu

TATJANA PETKOVIĆ, MIROSLAV ĆIRIĆ
UNIVERSITY OF NIŠ
FACULTY OF SCIENCES AND MATHEMATICS
DEPARTMENT OF MATHEMATICS
ĆIRILA I METODIJA 2, P.O. BOX 224
18000 NIŠ
YUGOSLAVIA

E-mail: tanjapet@pmf.pmf.ni.ac.yu, mciric@pmf.pmf.ni.ac.yu

TATJANA PETKOVIĆ
UNIVERSITY OF TURKU
TURKU CENTRE FOR COMPUTER SCIENCE
AND DEPARTMENT OF MATHEMATICS
FIN-20014 TURKU
FINLAND

E-mail: tatjana@cs.utu.fi

(Received August 9, 1999; accepted December 20, 1999)