On testing output faults in the McCluskey Fault Model

ULRIKE BRANDT
Technische Universität Darmstadt
e-mail: `brandt@informatik.tu-darmstadt.de`

and

HERMANN K.-G. WALTER
Technische Universität Darmstadt
e-mail: `walter@informatik.tu-darmstadt.de`

ABSTRACT

McCluskey et al. introduced a very general fault model for finite automata. In this paper we will show that all testable output faults can be tested by a single input word in this model. Furthermore, in the case of irreducible automata we will show that this is true for all output faults. Our main tool to prove the results is a careful analysis of the structure of automata especially considering subautomata and edge-(state)traverses of the transition graph induced by input words.

Keywords: finite automata, faults, testability, subautomata, irreducible automata, traverses

0. Introduction

In [1], [4] and following papers McCluskey et al. developed a very general fault model for finite automata. A fault of an automaton A is any different automaton with the same inputs, outputs and states. They considered various classes of faults and gave algorithms for calculating test sets, i.e. sets of input words so that the resulting outputs indicate whether a fault is present or not. Testing a fault can be done assuming that the automaton always starts with the same state (testability) or in (possibly) different states (strong testability). The latter one is closely related to the structure of the given automaton. This is also true for output faults – faults which affect only the output-unit. An output fault has the same transitions as the given automaton. We can show that for a given automaton the class of all its strongly testable output faults can be tested by a single word. Fault diagnosis is connected to experiments on automata first studied by Moore [3]. Though faults are not mentioned one can find a few remarks in [2] touching on this connection. Moreover, McCluskey et al. put some emphasis on reset mechanisms, though they are not part of the automaton and operate faultfree. In connection with traverses of edges respectively states we will make heavy use of resets. But we do not assume that they are an additional faultfree part of the automaton under consideration.

1. Basic Notations and Definitions

An *alphabet* X is a finite set of *letters*. The set of *words* (over X) is the free monoid X^* over X with the *empty* word $\Box$ as identity. If $w = x_1 \dots x_n$ ($x_i \in X$ for $1 \le i \le n$) the *length* of w is $|w| = n$. For $L_{1,2} \subseteq X^*$ the *complex product* is defined by $L_1L_2 = \{w_1w_2 | w_1 \in L_1 \text{ and } w_2 \in L_2\}$. We use the usual convention for singletons in identifying w with $\{w\}$, if no confusion is possible. X^* can be partially ordered by the *prefix* relation defined by

$$w \le v(\mathbf{pref}) \Longleftrightarrow v \in wX^*.$$

Definition 1.1 *A* (Mealy-)automaton *is a quadruple* $A = (I, S, O, \delta, \lambda)$ *where*

- *I and O are alphabets* (inputs *respectively* outputs)
- *S is a finite set* (states)
- *$\delta : I \times S \to S$ and $\lambda : I \times S \to O$ are the* transition *and* output *functions respectively.*

We extend δ and λ to words by the formulas below, where $w, v \in I^*, s \in S$

$$\delta(\Box, s) = s, \delta(wv, s) = \delta(v, \delta(w, s)) \quad (\textit{transition} \text{ formula) and}$$
$$\lambda(\Box, s) = \Box, \lambda(wv, s) = \lambda(w, s)\lambda(v, \delta(w, s)) \quad (\textit{response} \text{ formula).}$$

For letters these extensions are the given δ and λ. Fixing $s \in S$ as a starting state we define the *(realized)* function $\lambda^s(w) = \lambda(w, s)$ for $w \in I^*$. Note that $|\lambda^s(w)| = |w|$ always holds. An automaton A is *minimal* if and only if

$$\forall s, s' \in S : \lambda^s = \lambda^{s'} \Rightarrow s = s'.$$

For two automata A and A' with $I = I'$ and $O = O'$ a mapping $\phi : S \to S'$ is a *homomorphism* if and only if

$$\forall x \in I, s \in S : \delta'(x, \phi(s)) = \phi(\delta(x, s)) \text{ and } \lambda'(x, \phi(s)) = \lambda(x, s).$$

If a homomorphism ϕ is given, it is easy to prove that $\lambda^s = \lambda'^{\phi(s)}$ for all $s \in S$. A bijective ϕ is an *isomorphism* and we write $A \cong A'$ or $A \cong_\phi A'$. Note that ϕ^{-1} is also an isomorphism.

Another model for a device with finite memory is the *Moore-automaton*; it is a Mealy-automaton where the output function is given by

$$\lambda(x, s) = \mu(\delta(x, s)) \quad (x \in I, s \in S)$$

with a function $\mu : S \to O$ (*marking*). Our constructions will considerably simplify, if we use Moore-automata. This will be discussed at the proper places.

In the following we will fix I and O and denote the collection of all automata with state set S by $\mathbf{Autom}(S)$. A *fault* for $A \in \mathbf{Autom}(S)$ is any $A_f \in \mathbf{Autom}(S)$ with $A_f \ne A$. By symmetry A is then a fault of A_f. We use the subscript "$_f$" to denote the fault automaton. We shall discuss two forms of testability. In the weaker one the test is started with the same initial state for both automata. In the stronger form they may start in different states.

Definition 1.2 *Let $A, A_f \in \mathbf{Autom}(S)$ and $s \in S$.*

- *A_f is s-testable (for A) if $\lambda^s \ne \lambda_f^s$.*

- *A_f is strongly s-testable (for A) if $\lambda^s \neq \lambda_f^{s'}$ for all $s' \in S$.*

We extend this definition to subsets $S' \subseteq S$ by calling A_f (*strongly*) *S'-testable* (for A) if A_f is (strongly) s-testable for every $s \in S'$. A_f is (*strongly*) *testable* for A if it is (strongly) S-testable for A.
Note that in the case $\#(O) = 1$ no A_f is testable on nonempty S'. We assume for the following $\#(O) > 1$. Strong testability respects isomorphisms in the sense that A_f is not strongly testable on any nonempty subset of S if $A_f \cong A$. This is not true for testability. Fig. 1.1 shows such a pair of isomorphic automata which are both testable for each other. In this situation testability depends on the isomorphism.

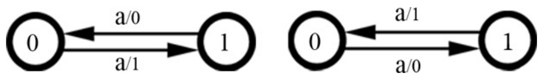

Figure 1.1 Isomorphisms and faults

Observation 1.3 *Let $A, A_f \in$ **Autom**(S) with $A \cong A_f$ and A minimal. A_f is testable for A if and only if there exists an isomorphism ϕ with $A_f \cong_\phi A$ and $\phi(s) \neq s$ for all $s \in S$, i.e. ϕ must be free of fixpoints.*

Proof. If $s \in S$ exists with $\phi(s) = s$ then $\lambda^s = \lambda_f^s$ and A_f is not s-testable. Conversely, suppose there exists $s \in S$ with $\lambda_f^s = \lambda^s$. We know $\lambda^{\phi(s)} = \lambda_f^s$, hence $\lambda^s = \lambda^{\phi(s)}$. Since A is minimal $s = \phi(s)$ and we find a fixpoint – a contradiction. □

The second extension of our definitions deals with fault classes. For a given $A \in \mathbf{Autom}(S)$ a *fault class* $\mathscr{F}$(for A) is a subset of $\mathbf{Autom}(S)$ with $A \notin \mathscr{F}$. Such fault classes are mainly derived from a common schema of automata – both for Mealy- and Moore-type presented in fig. 1.2. Typical faults can affect the logical units Transition and Output, the Memory and the connections between these components.

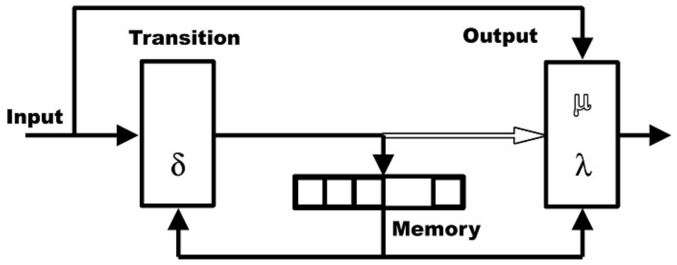

Figure 1.2 Basic structure of automata, Mealy: dotted line, Moore: hollow line

We get for example $\mathscr{F}_{\mathbf{out}}(A) = \{A_f \in \mathbf{Autom}(S) \mid \delta = \delta_f \text{ and } \lambda \neq \lambda_f\}$ as the class of *output faults*. For a Moore-automaton A we can single out $\mathscr{F}_{\mathbf{mout}} = \{A_f \in$

$\mathscr{F}_{\mathbf{out}}|A_f$ is a Moore-automaton$\}$. Note that an output fault A_f of a Moore-automaton does not need to be a Moore-automaton.

For $S' \subseteq S$ such a fault class $\mathscr{F}$ is (*strongly*) S'*-testable* (for A) if every $A_f \in \mathscr{F}$ is (strongly) S'-testable. If $\mathscr{F}$ is (strongly) S-testable then $\mathscr{F}$ is (*strongly*) *testable* for A.

We turn our interest to fault-testing. Consider $A \in \mathbf{Autom}(S)$, $S' \subseteq S$ and a fault class $\mathscr{F}$ of A. A set $T \subseteq I^*$ is a S'*-test* for A and $\mathscr{F}$ if and only if

$$\forall\, A_f \in \mathscr{F},\ s\ \in S',\ \lambda^s \neq \lambda_f^s\ \exists\, w \in T : \lambda^s(w) \neq \lambda_f^s(w)$$

Since $\mathbf{Autom}(S)$, I and O are all finite sets a finite S'-test T always exists for a given A and $\mathscr{F}$. Note, that $\mathscr{F}$ need not be S'-testable. T is a *strong* S'*-test* for A and $\mathscr{F}$ if and only if

$$\forall\, A_f \in \mathscr{F}, s \in S,\ s' \in S'\ \exists\, w \in T : \lambda^{s'}(w) \neq \lambda_f^s(w).$$

In this case, a strong (finite) S'-test exists if and only if $\mathscr{F}$ is strongly S'-testable for A. A strong S'-test is always a S'-test. As before, T is a (*strong*) *test* (for A and $\mathscr{F}$) if and only if T is a (strong) S-test. In connection with tests the following fact is fundamental.

Observation 1.4 *If* $A,\ A_f \in \mathbf{Autom}(S)$ *and* $s,\ s' \in\ S$ *then for all* $v,\ w \in I^*$

$$\lambda^s(vw) = \lambda_f^{s'}(vw) \Leftrightarrow (\lambda^s(v) = \lambda_f^{s'}(v) \text{ and } \lambda^{\delta(v,s)}(w) = \lambda_f^{\delta_f(v,s')}(w)).$$

Proof. By the response formula we know

$$\lambda(vw,s) = \lambda^s(v)\lambda^{\delta(v,s)}(w) \text{ and } \lambda_f^{s'}(vw) = \lambda_f^{s'}(v)\lambda_f^{\delta_f(v,s')}(w).$$

If $\lambda^s(vw) = \lambda_f^{s'}(vw)$ then $\lambda^s(v) = \lambda_f^{s'}(v),$ since $|\lambda^s(v)| = |\lambda_f^{s'}(v)|$. But then by cancellation $\lambda^{\delta(v,s)}(w) = \lambda_f^{\delta_f(v,s')}(w)$ □

For example, we directly obtain from this fact, that $(T\backslash w)\ \cup\ ww'$ is a (strong) S'-test for any (strong) S'-test T, $w' \in\ I^*$ and $w \in\ T$.

Example 1.5 (*reset automata*) Consider an alphabet X and $\mathbb{B} = \{0,1\}$. The (*letter-*)*reset* automaton $\mathrm{Res}_X = (X,X,\mathbb{B},\delta,\lambda)$ is given by

$$\delta(x,y) = x \text{ and } \lambda(x,y) = \boldsymbol{\delta}_{x,y} \qquad (\boldsymbol{\delta} : \text{Kronecker symbol}) \quad (x,y \in X).$$

The fault class is given in the following way. Consider a mapping $\sigma : X \to X$. Then define $\mathrm{Res}_{\sigma,X} = (X,X,\mathbb{B},\delta_\sigma,\lambda)$ with $\delta_\sigma(x,y) = \sigma(x)(x,y \in X)$ leaving λ unchanged. The fault class is under consideration the collection of all these automata where σ is not the identity on X. This fault class includes quite typical faults considering fig. 1.2. For example some of the register cells may be stuck at a certain value. Note that cuts in the connections may also cause such faults.

Now for a given fault $A_f = \mathrm{Res}_{\sigma,X}$ we use input words of the form $w = xx$ $(x \in X)$. We get for all $y \in X : \lambda(xx,y) = \boldsymbol{\delta}_{x,y}1$ and $\lambda_f(xx,y) = \boldsymbol{\delta}_{x,y}\boldsymbol{\delta}_{x,\sigma(x)}$. Since σ is not the identity on X a $x \in X$ exists with $\sigma(x) \neq x$. But now xx tests this fault. This shows that $\{xx \mid x \in X\}$ is a test for Res_X and this fault class. Moreover, it is a strong test. In the next section we shall prove that this fact is true in a more general setting, where the automaton Res_X serves as an example. □

Our second example deals with a more refined look at faults.

Example 1.6 (*switching circuits*) We consider realizations of automata by switching circuits.

If we follow the standard schema for automata, the transition and output unit are usually realized by switching circuits without feedbacks. To do this we need a binary encoding of all three sets – inputs, outputs and states. Then transition and output become boolean functions which can be realized by switching circuits without feedbacks. We make free use of a basic system of gates containing the boolean functions $x \cdot y$ (AND), $x + y$ (OR), $x \oplus y$ (EXOR) and their complementations $x \cdot^c y$ (NAND), $x +^c y$ (NOR) and $x \oplus^c y$ (NEXOR). Now consider a four letter alphabet $X = \{a, b, c, d\}$ and Res_X. Encoding of the letters is given by

a	b	c	d
00	01	10	11.

If we use the encodings $x_1 x_2$ for the letter x and $s_1 s_2$ for the state s we see the output $z = (x_1 \oplus s_1) +^c (x_2 \oplus s_2)$. The reader should bear in mind that EXOR tests for inequality of bits. The corresponding switching circuit is shown in fig. 1.3 together with three faults – a cut, a contact and their combination.

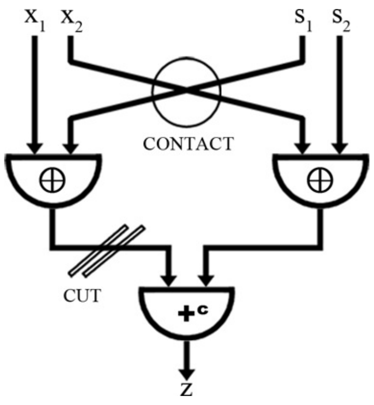

Figure 1.3 Fault testing

We assume that the hardware implementation of this circuit results for the cut in a stuck-at-0-fault and for the contact in an additional OR-gate. The resulting output functions in the presence of these faults are

$$z_{\mathrm{Cut}} = x_2 \oplus^c s_2,\ z_{\mathrm{Contact}} = (x_1 \oplus (x_2 + s_1)) +^c (s_2 \oplus (x_2 + s_1)),\ z_{\mathrm{Comb}} = (x_2 + s_1) \oplus^c s_2.$$

Decoding yields for example

$$\lambda_{\mathrm{Cut}}(x, s) = (\boldsymbol{\delta}_{x,a} + \boldsymbol{\delta}_{x,c}) \cdot (\boldsymbol{\delta}_{a,s} + \boldsymbol{\delta}_{c,s}) + \big(\boldsymbol{\delta}_{x,b} + \boldsymbol{\delta}_{x,d}\big) \cdot \big(\boldsymbol{\delta}_{b,s} + \boldsymbol{\delta}_{d,s}\big)\,.$$

The following table summarizes the values for all possible inputs.

Output pattern $z\ z_{cut}\ z_{contact}\ z_{comb}$	a	b	c	d
a	1111	0000	0101	0000
b	0000	1101	0000	0111
c	0100	0000	1100	0000
d	0001	0101	0011	1111

Inspection of this table yields possible tests for these faults together with the corresponding tests for the faulty automation, as shown in the next table.

	Fault tests	
Cut	Contact	Comb
00 10 *ca*	01 01 *bb*	00 11 *da*
01 11 *db*	10 10 *cc*	01 11 *db*
10 00 *ac*	10 11 *dc*	10 00 *ac*
11 01 *bd*	11 01 *bd*	10 10 *cc*
		10 11 *dc*
		11 01 *bd*

The test 1101 is the only test for all three faults. It corresponds to the input *bd*, i.e. first the input *b* sends Res_x to the state b and then use the input *d* for fault detection. □

2. Subautomata and Testability

In dealing with testability it is quite reasonable to study for an automaton $A \in \mathbf{Autom}(S)$ and a fault A_f the set $\mathbf{nontest}(A, A_f) = \{s \in S \mid \exists s' \in S \colon \lambda^s = \lambda_f^{s'}\}$.

Observation 2.1 *For any two automata $A, A_f \in \mathbf{Autom}(S)$:*

$$\forall x \in I : s \in \mathbf{nontest}(A, A_f) \Rightarrow \delta(x,s) \in \mathbf{nontest}(A, A_f).$$

Proof. Let $s, s' \in S$ with $\lambda^s = \lambda_f^{s'}$. Applying fact 1.6 for $x \in I$ and $w \in I^*$ we find $\lambda^{\delta(x,s)} = \lambda_f^{\delta_f(x,s')}$, i.e. $\delta(x,s) \in \mathbf{nontest}(A, A_f)$. □

Definition 2.2 *Let $S' \subseteq S, A \in \mathbf{Autom}(S)$ and $A' \in \mathbf{Autom}(S')$. A' is a* subautomaton *of $A(A' \in \mathbf{sub}(A))$ if and only if $\delta'(x,s) = \delta(x,s)$ and $\lambda'(x,s) = \lambda(x,s)$ for all $x \in I$ and $s \in S'$.*

If $A' \in \mathbf{sub}(A)$ then for any $x \in I$ and $s \in S'$: $\delta(x,s) \in S'$. Conversely, if $S' \subseteq S$ satisfies this condition then a subautomaton with state set S' is defined by the restriction of δ and λ to $I \times S'$. Therefore we do not distinguish subautomata and subsets of S satisfying this condition. In this way we get $\varnothing, S \in \mathbf{sub}(A)$. By obs. 2.1 $\mathbf{nontest}(A, A_f) \in \mathbf{sub}(A)$ for all

$A, A_f \in \mathbf{Autom}(S)$. There are some canonical subautomata of $A \in \mathbf{Autom}(S)$. To any $s \in S$ we can consider

$$\mathbf{reach}(s,A) = \{\delta(w,s) \mid w \in I^*\}.$$

By the transition formula for any $x \in I$ and $s' = \delta(w,s)$: $\delta(x,s') = \delta(x,\delta(w,s)) = \delta(wx,s)$. This proves $\mathbf{reach}(s,A) \in \mathbf{sub}(A)$. **reach** is a consistent monotonic operation, that means $s \in \mathbf{reach}(s,A)$ and $\mathbf{reach}(\delta(w,s),A) \subseteq \mathbf{reach}(s,A)$ for all $s \in S$ and $w \in I^*$. We want to apply obs. 2.1 to automata which have no nontrivial subautomata.

Definition 2.3 *$A \in \mathbf{Autom}(S)$ is* irreducible *if and only if $\mathbf{reach}(s,A) = A$ for all $s \in S$.*

Equivalently, an automaton $A \in \mathbf{Autom}(S)$ is irreducible if and only if a function $\mathbf{reset}\colon S \times S \to I^*$ exists with: $\delta(\mathbf{reset}(s,s'),s) = s'$ for all $s,s' \in S$.

If S is a singleton or empty then A is always irreducible. The automata Res_X are irreducible with the function $\mathbf{reset}(y,y') = y'(y,y' \in X)$.

Another way to define irreducibility is given by the following

Observation 2.4 *$A \in \mathbf{Autom}(S)$ is irreducible if and only if for all $S' \in \mathbf{sub}(A) : S' \neq \emptyset \Rightarrow S' = S$.*

Lemma 2.5 *If $A \in \mathbf{Autom}(S)$ is minimal and irreducible then $A_f \in \mathbf{Autom}(S)$ is strongly testable for A if and only if $A \not\cong A_f$.*

Proof. Suppose $A \not\cong A_f$ and A_f is not strongly testable for A. But then $\mathbf{nontest}(A,A_f) \neq \varnothing$ and therefore by Observation 2.4: $\mathbf{nontest}(A,A_f) = S$. By symmetry $\mathbf{nontest}(A_f,A) \neq \varnothing$, too. Define $\psi\colon \mathbf{nontest}(A_f,A) \to S$ by $\psi(s') = s$ with $\lambda_f^{s'} = \lambda^s$. Since A is minimal ψ is well-defined. Since $S = \mathbf{nontest}(A,A_f)$, ψ is surjective, and therefore ψ is a bijection. By this $\mathbf{nontest}(A_f,A) = S$. Moreover, ψ is obviously a homomorphism. In total ψ is an isomorphism and we get a contradiction. The reverse direction is trivial. □

For an automaton A the irreducible subautomata play an important role. We introduce

$$\mathbf{bottom}(A) = \{s \in S \mid \forall s' \in \mathbf{reach}(A,s)\colon s \in \mathbf{reach}(A,s')\}.$$

If S is not empty, $\mathbf{bottom}(A)$ is also not empty. This can be seen easily by looking at the set system $\{\mathbf{reach}(A,s) \mid s \in S\}$. This system is partially ordered by inclusion. Since S is finite we find $\mathbf{bottom}(A)$ as the union of the minimal elements. Clearly, $\mathbf{bottom}(A) \in \mathbf{sub}(A)$. If $A' \in \mathbf{sub}(A)$ with state sets, then for all $s \in S' \mathbf{reach}(A,s) = \mathbf{reach}(A',s)$, hence $\mathbf{bottom}(A') \in \mathbf{sub}(\mathbf{bottom}(A))$. Moreover, for any $s \in \mathbf{bottom}(A)$ $\mathbf{reach}(A,s)$ is by definition irreducible. Conversely, if $A' \in \mathbf{sub}(A)$ is irreducible then $A' \in \mathbf{sub}(\mathbf{bottom}(A))$.

Observation 2.6 *If $A \in \mathbf{Autom}(S)$ then there exists a function $\mathbf{down}\colon S \to I^*$ with*

$$\forall s \in S\colon \delta(\mathbf{down}(s),s) \in \mathbf{bottom}(A).$$

Proof. We use induction on $\#(S)$. If $\#(S)=1$, $A=\mathbf{bottom}(A)$ and the empty word will do. Let $\#(S)>1$. If $\mathbf{bottom}(A)=A$ we can use again the empty word. Suppose $\mathbf{bottom}(A)\neq A$. Fix $s\notin\mathbf{bottom}(A)$. Then $s'\in\mathbf{reach}(A,s)$ exists with $s\notin\mathbf{reach}(A,s')$. Consider the set $S'=\{s'\in S\mid s'\in\mathbf{reach}(A,s)$ and $s\notin\mathbf{reach}(A,s')\}$ then $\#(S')<\#(S)$, S' is not empty and S' defines a subautomaton A' of A. By induction hypothesis we find for any $s'\in S'$ $\mathbf{down}_{A'}(s')$ with $\delta(\mathbf{down}_{A'}(s'),s')\in\mathbf{bottom}(A')\in\mathbf{sub}(\mathbf{bottom}(A))$. Moreover, for such a s' exists a word $w\in I^*$ with $\delta(w,s)=s'$ and we can define $\mathbf{down}_A(s)=w\mathbf{down}_{A'}(s')$. Using the transition formula we get

$$\delta(w\mathbf{down}_{A'}(s'),s)=\delta(\mathbf{down}_{A'}(s'),\,\delta(w,s))=\delta(\mathbf{down}_{A'}(s'),s')\in\mathbf{bottom}(A). \qquad \square$$

If $\mathbf{d}:S\to I^*$ satisfies for $s\in S$ $\delta(\mathbf{d}(s),s)\in\mathbf{bottom}(A)$, then for any $w\in I^*$ $\delta((\mathbf{d}(s)w),s)\in\mathbf{bottom}(A)$. Hence, there are infinitely many choices for $\mathbf{down}_A$.

Lemma 2.7 *Let $A,A_f\in\mathbf{Autom}(S)$, then A_f is strongly testable for A if and only if A_f is strongly $\mathbf{bottom}(A)$-testable for A.*

Proof. Consider $s,s'\in S$.
If A_f is strongly $\mathbf{bottom}(A)$-testable for A there exists $w_0\in I^*$ with

$$\lambda^{\delta(\mathbf{down}_A(s,),s)}(w_0)\neq\lambda_f^{\delta_f(\mathbf{down}_A(s,),s')}(w_0).$$

But then by fact 1.6 $\lambda^s(\mathbf{down}_A(s)w_0)\neq\lambda_f^s(\mathbf{down}_A(s)w_0)$. $\square$

We can strengthen obs 2.6 in such a way that the words $\mathbf{down}_A(s)$ can be combined to a single word sending all states to $\mathbf{bottom}(A)$.

Theorem 2.8 *To any $A\in\mathbf{Autom}(S)$ there exists $\mathbf{down}(A)\in I^*$ such that*

$$\forall s\in S\colon\delta(\mathbf{down}(A),s)\in\mathbf{bottom}(A).$$

Proof. We use a $\mathbf{down}(s)$ as in obs. 2.6. Numbering the states of S from $1,\dots,m$ we use the following programming piece to construct $\mathbf{down}(A)$:

```
down(A):
  $w := \square$
  for $i := 1; i \le m; i = i+1$ do
    $w := w\,\mathbf{down}\delta(w,s_i)$
  end for
  return w
```

If $1\le i\le m$ we can decompose $\mathbf{down}(A)=u\mathbf{down}\delta(u,s_i)v$ with suitable $u,v\in I^*$. But then $\delta(\mathbf{down}(A),s_i)=\delta(v,\delta(\mathbf{down}(\delta(u,s_i)),\delta(u,s_i)))$. We know

$$\delta(\mathbf{down}(\delta(u,s_i)),\delta(u,s_i))\in\mathbf{bottom}(A)$$

and $\delta(v,s)\in\mathbf{bottom}(A)$ for all $s\in\mathbf{bottom}(A)$. By this $\delta(\mathbf{down}(A),s_i)\in\mathbf{bottom}(A)$. $\square$

Example 2.9 *Construction of* **down**(A). For k > 0 consider the Moore-automata A_k with $I = \{a,b\}$, $O = \mathbb{B}$, S_k = [0 .. 2k + 1] and δ_k, μ_k defined by

$$\delta_k(a,0) = \delta_k(b,0) = \delta_k(b,1) = \delta_k(a,2) = 0,$$
$$\delta_k(a,2i+1) = 2(i+1)+1(0 \leq i < k), \delta_k(a,2i) = 2(i-1)(1 < i \leq k), \delta_k(a,2k+1) = 2k,$$
$$\delta_k(b,2i+1) = 2(i-1)+1(0 < i \leq k), \delta_k(b,2i) = 2(i+1)(1 \leq i < k), \delta_k(b,2k) = 2k+1,$$
$$\mu_k(j) = j \,\mathbf{mod}\, 2 \ (j \in S_k).$$

Clearly, **bottom**(A_k) has only one state, namely 0. A possible $\mathbf{down}_A$ is given by

$$\mathbf{down}_A(0) = \square, \mathbf{down}_A(2i+1) = b^{i+1}(0 \leq i \leq k), \mathbf{down}_A(2i) = a^{i+1}(1 \leq i \leq k).$$

Consider the construction in the proof of th. 2.7 starting with state 0. The outcome depends on the choices made during this construction. Choosing the odd numbers first in ascending order and then the even ones in descending order yields **down**(A) = b^{2k+1}. Proceeding the other way round - first even, then odd ones - yields **down**(A) = a^{2k+1}. □

We get the following with respect to tests. If $T \subseteq I^*$ is a strong **bottom**(A)-test for A and A_f then **down**$(A)T$ is a strong test for A and A_f. So the size of the test remains unchanged. We mentioned above that **bottom**(A) is the union of the irreducible subautomata of A.
With respect to fault testing we refine this observation by introducing the *direct sum* of automata.
Let $A_{1,2} \in \mathbf{sub}(A)$ with state sets $S_{1,2}$. If $S_1 \cap S_2 = \varnothing$ and $S_1 \cup S_2 = S$, A is the *direct sum* of A_1 and A_2 ($A = A_1 \oplus A_2$). The direct sum is associative and commutative and we extend it to finitely many $A_i (1 \leq i \leq k)$ obtaining $A = A_1 \oplus \cdots \oplus A_k$ in the usual way.

Lemma 2.10 *For any* $A \in \mathbf{Autom}(S)$ $\mathbf{bottom}(A) = A_1 \oplus \cdots \oplus A_k$ *where all* A_i *are irreducible. Moreover, this decomposition is unique up to permutations of the components.*

Proof. We know that any irreducible $A' \in \mathbf{sub}(A)$ is of the form $A' = \mathbf{reach}(A,s)$ for some $s \in S$. Let $s' \in S$. If there exists $s'' \in \mathbf{reach}(A,s) \cap \mathbf{reach}(A,s')$, then $s \in \mathbf{reach}(A,s'') \subseteq \mathbf{reach}(A,s')$, and therefore by irreducibility $\mathbf{reach}(A,s)$ = $\mathbf{reach}(A,s')$

But then consider a system $s_1,\ldots,s_k$ of states with

- $\mathbf{reach}(A,s_i)$ irreducible, $\mathbf{reach}(A,s_i) \cap \mathbf{reach}(A,s_j) = \varnothing \quad (1 \leq i \neq j \leq k)$ and
- $\forall s \in S, \mathbf{reach}(A,s)$ irreducible $\exists 1 \leq i \leq k : s \in \mathbf{reach}(A,s_i)$.

Now the $\mathbf{reach}(A,s_i)$ constitute the desired decomposition with $A_i = \mathbf{reach}(A,s_i)$. □

3. Edge- and state-traverses

Traversing the transitions is an important tool to study testability and design tests. Such traverses are quite familiar in graph-theory. In our case an additional feature is present. The traverse must be triggered by inputs.
Therefore we define the following two functions $\mathbf{evisit}_A : I^* \times S \to \mathbf{2}^{I \times S}$ and $\mathbf{svisit}_A : I^* \to \mathbf{2}^S$ by

$$\mathbf{evisit}_A(w,s) = \{(x,s') \in I \times S \,\big|\, \exists u \in I^* : ux \leq w(pref) \text{ and } \delta(u,s) = s'\}.$$

and

$$\mathbf{svisit}_A(w,s) = \{s' \in S \,|\, \exists u \in I^* : u \neq \Box, u \leq w(pref) \text{ and } \delta(u,s) = s'\}(w \in I^*, s \in S).$$

Definition 3.1 *Let $A \in$* **Autom**(S)*, $s \in S$ and $w \in I^*$. w is an* edge-traverse *for s, if* **evisit**$_A$ *(w, s) = $I \times S$, and a* state-traverse *for s, if* **svisit**$_A(w,s)$ *= S.*

We call $w \in I^*$ an *edge-(state-)traverse* of A if w is an edge-(state-)traverse for all $s \in S$. Clearly, an edge-traverse for s is always a state-traverse for s. If a state-traverse exists, A must be irreducible.

Example 3.2 We study again the automata A_k from ex. 2.8. We calculate

$$\delta_k(b^i, 2i+1) = \delta_k(b^{k+i}, 2(k-i+1)) = 1 (0 \leq i \leq k).$$

Then aba^{2k+1} is a state-traverse for 1, $b^i a^{2k+1}$ is a state-traverse for 2i + 1 (i >0) and b^{k+i} a^{2k+1} a state-traverse for 2(k - i + 1)(1 $\leq$ i $\leq$ k). For 0 no state-traverse exists. Suppose $s \in S_k$ exists such that an edge-traverse w for s can be found. But then both (b, 1) and (a, 2) are elements of **evisit**$_{A_k}$ (w,s). Since $\delta_k(a,0)$ = $\delta_k(b,0)$ = 0 this is impossible. Hence, for no state $s \in S_k$ an edge-traverse can be found. Look for example at state 1. Then **evisit**$_{A_k}$ $(a^{2k}b^{2k+1}ab, 1)$ = $\{a, b\}$ $\times S_k \setminus (a,2)$. □

Lemma 3.3 *Let $A \in$* **Autom**(S)*, $s \in S$ and $w \in I^*$. Then w is an edge-traverse for s if and only if $\lambda^s(w) \neq \lambda^s_f(w)$ for all $A_f \in \mathscr{F}_{\mathbf{out}}(A)$.*

Proof. Consider $A_f \in \mathscr{F}_{\mathbf{out}}(A)$. Then there exists $(x,s') \in I \times S$ with $\lambda(x,s') \neq \lambda_f(x,s')$. Since w is an edge-traverse for $s, w = uxv$ with $\delta(u,s) = s'$. By fact 1.6 we get $\lambda^s(ux) \neq \lambda^s_f(ux)$ and then $\lambda^s(w) = \lambda^s(uxv) \neq \lambda^s_f(uxv) = \lambda^s_f(w)$. Hence w is a s-test.
Conversly, assume w is not an edge-traverse for s. Then $(x,s') \in I \times S$ exists with (x, s') $\notin$ **evisit**$_A(w,s)$. Consider the fault $A_f \in \mathscr{F}_{\mathbf{out}}(A)$ given by $\lambda_f(x,s'') = \lambda(x,s'')$ if $s'' \neq s'$ and $\lambda_f(x,s') \neq \lambda(x,s')(x \in I, s'' \in S)$ $(\#(O) > 1!)$. Then $\lambda^s(w) = \lambda^s_f(w)$ - a contradiction. Hence w must be an edge-traverse for s. □

We get a lower bound for the length of those $w \in I^*$ which are edge-traverses for s on A, because we have to meet every pair (x,s'). If $n = \#(X)$ and $\#(S) = m$ then $|w| \geq nm$.

With little changes the same result is true for Moore-automata A and $\mathscr{F}_{\mathbf{mout}}(A)$ and state-traverses.

Lemma 3.4 *Let $A \in$* **Autom**(S) *a Moore-automaton, and $w \in I^*$. Then w is a state-traverse for s if and only if $\lambda^s(w) \neq \lambda^s_f$ for all $A_f \in \mathscr{F}_{\mathbf{mout}}(A)$*

Proof. Consider $A_f \in \mathscr{F}_{\mathbf{mout}}(A)$. Then there exists $s' \in S$ with $\mu(s') \neq \mu_f(s')$. Since w is a state-traverse for $s, w = uxv$ with $\delta(ux,s) = s'$. Note $\lambda(ux,s) = \lambda(u,s)\mu(s') \neq \lambda_f(u,s)\mu_f(s') = \lambda_f(ux,s)$. Again by fact 1.6 $\lambda^s(w) \neq \lambda^s_f(w)$.

Conversly, assume w is not a state-traverse for s. Then $s' \in S$ exists with $s' \notin$ **svisit**$_A(w,s)$. Consider the fault $A_f \in \mathscr{F}_{\mathbf{mout}}(A)$ given by $\mu_f(s'') = \mu(s'')$ if $s'' \neq s'$ and $\mu_f(s') \neq \mu(s')(x \in I, s'' \in S)$. Then $\lambda^s(w) = \lambda^s_f(w)$ - a contradiction. Hence w must be a state-traverse for s. □

For a state-traverse w we clearly have the lower bound $|w| \geq m$. Note at this point that turning a Mealy-automaton into an equivalent Moore-automaton changes the character of the fault. An output fault becomes a transition fault.

Applying the transition formula we get straight forward that for an edge-(state-)traverse w of s and for any $v \in I^*$ wv is an edge-(state-)traverse for s, too. Moreover, if w is an edge-(state-)traverse for A then for any $u, v \in I^*$ uwv is also an edge-(state-)traverse of A.
The existence of such traverses for irreducible automata is asserted by the following two results where the proofs additionally exhibit algorithms to find these traverses. We start with the easier task determining state-traverses.

Lemma 3.5 *For any irreducible $A \in \mathbf{Autom}(S)$ a state-traverse for A exists.*

Proof. We use a function **reset** with $\mathbf{reset}(s,s) \neq \Box$ for $s \in S$ associated to (the irreducible) A and number the set of states $S = \{s_1, \ldots, s_m\}$. Consider the word $\mathbf{start} = \mathbf{reset}(s_1, s_2) \ldots \mathbf{reset}(s_{m-1}, s_m)$. If $s \in S$ and $u \in I^*$ then the word $u\mathbf{reset}(\delta(u,s), s_1)\mathbf{start}$ is always a state-traverse for s. We obtain a state-traverse of A by the following programming piece which uses the functions $\mathbf{trav}(w) = \{s \in S \mid w \text{ is a state-traverse for } s\} (w \in I^*)$

```
strav(A):
  w := $\Box$
  trav := $\varnothing$
  while trav $\subset S$ do
    choose($s \in S \setminus$ trav)
    w := $w\mathbf{reset}(\delta(w,s), s_1)\mathbf{start}$
    trav := trav(w)
  end while
  return w
```
$\Box$

We get the following worst-case estimate for $|\mathbf{strav}(A)|$. Clearly, we can choose $\mathbf{reset}(s,s')$ such that $|\mathbf{reset}(s,s')| \leq m-1$. Then $|\mathbf{start}| \leq (m-1)^2$. Suppose we can eliminate only one state in each turn of the loop then there are m such turns. In total $|\mathbf{strav}(A)| \leq m(m-1)^2$. It is not surprising that this upper bound is independent of the numbers of inputs.
We refine this construction to get an edge-traverse visiting all outgoing edges of a state s if we meet s during a state-traverse.

Theorem 3.6 *For any irreducible $A \in \mathbf{Autom}(S)$ there exists an edge-traverse.*

Proof. Consider again a function $\mathbf{reset}(s,s')$ for A. Again let $S = \{S_{1,}, S_m\}$ furthermore let $I = \{x_1, \ldots, x_n\}$. Define a function $\mathbf{edge}\colon S \to I^*$ by

$$\mathbf{edge}(s) = x_1\mathbf{reset}(\delta(x_1,s),s) \ldots \mathbf{reset}(\delta(x_{n-1},s)x_n.$$

$\mathbf{edge}(s)$ traverses all outgoing edges of s if we perform $\delta(\mathbf{edge}(s),s)$, i.e. for all $x \in I$ there exists $u \in I^*$ with $ux \leq \mathbf{edge}(s)(\mathbf{pref})$. This time consider the word $\mathbf{start} = \mathbf{edge}(s_1)\mathbf{reset}(s_1,s_2)\mathbf{edge}(s_2)...\mathbf{reset}(s_m-1, s_m-2)\mathbf{edge}(s_m)$.

We use the function $etrav(w) = \{s \in S \mid w$ is an edge-traverse for $s\}$ for the following programming piece:

```
etrav(A):
  $w := \Box$
  etrav $= \varnothing$
  while etrav $\subset S$ do
    choose($s \in S \setminus$ etrav)
    $w := w\,\mathbf{reset}(\delta(w,s), s_1)$start
    etrav := etrav($w$)
  end while
  return $w$
```

□

A worst case estimate can be derived as follows. As before $|\mathbf{reset}(s,s')| \leq m-1$. Then

$$|\mathbf{edge}(s)| \leq (n-1)(m-1)+1,$$

and then

$$|\mathbf{start}| \leq m((n-1)(m-1)+1)+(m-1)m = nm(m-1)+m$$

In the worst case only one state is eliminated at each turn of the loop. At each turn $|\mathbf{reset}(\delta(w,s),s_1)start|$ is added to $|w|$. Since there are m such turns we obtain $|\mathbf{strav}(A)| \leq m^2(n(m-1)+1)$.

Example 3.7 *(edge- and state-traverses)* Consider for n > 0 $X = \{x_1,\ldots,x_n\}$ and Res_X. We use the function $\mathbf{reset}(y,x) = x (x,y \in X)$. Then the construction given for state-traverses yields $\mathbf{start} = x_2 \ldots x_n$ and starting with state x_1 after the first run of the loop $w = x_1 \ldots x_n$. Now $\mathbf{trav}(w) = S$. Hence, no further loop follows and we obtain $\mathbf{strav}(\mathrm{Res}_X) = x_1 \ldots x_n$.

Turning to edge-traverses the construction of th.3.6 gives for $s \in S$ $\mathbf{edge}(s) = x_1 s x_2 \ldots s x_n$. There are n runs of the loop. We obtain $\mathbf{etrav}(\mathrm{Res}_X) = x_1\mathbf{edge}(x_1)x_2\mathbf{edge}(x_2)\ldots x_n\mathbf{edge}(x_n)$. Note that $|\mathbf{etrav}(\mathrm{Res}_X)| = 2n^2$. □

4. Output faults

We are now in the position to deal with testing output faults. In other words we design tests for $A \in \mathbf{Autom}(S)$ and $\mathscr{F}_{\mathbf{out}}(A)$, respectively $\mathscr{F}_{\mathbf{mout}}(A)$ in case A is a Moore-automaton.

Theorem 4.1 *If $A \in \mathbf{Autom}(S)$ is irreducible then $\mathscr{F}_{\mathbf{out}}(A)$ is strongly testable for A and any edge-traverse is a test for $\mathscr{F}_{\mathbf{out}}(A)$ and A. If A is a Moore-automaton, then any state-traverse is a test for $\mathscr{F}_{\mathbf{mout}}(A)$ and A.*

Proof. By th.3.6 an edge-traverse w for A exists. By le.3.2 $\mathscr{F}_{\mathbf{out}}(A)$ is testable and w is a test for $\mathscr{F}_{\mathbf{out}}(A)$. □

Next we consider direct sums of irreducible automata.

Lemma 4.2 *If $A \in \mathbf{Autom}(S)$ with $A = \mathbf{bottom}(A)$ then a $w \in I^*$ exists such that w is a test for A and $\mathscr{F}_{\mathbf{out}}(A)$.*

Proof. We know $A = A_1 \oplus \cdots \oplus A_k$ where $A_i \in \mathbf{Autom}(S_i)$ is irreducible for $1 \leq i \leq k$. Choose an edge-traverse w_i for any A_i. Let $w = w_1 \dots w_k$. Then w is an edge-traverse for A and all A_i. If $A_f \in \mathscr{F}_{\mathbf{out}}(A)$ then $\mathbf{bottom}(A_f) = A_f$. Since $\delta = \delta_f$ we get the corresponding decomposition $A_f = A_{f1} \oplus \cdots \oplus A_{fk}$ where $A_i = A_{fi}$ or $A_{fi} \in \mathscr{F}_{\mathbf{out}}(A_i)$ $(1 \leq i \leq k)$. Consider $s \in S$ with $\lambda^s \neq \lambda^s_f$ then for some $1 \leq i \leq k : s \in S_i$. In this case $A_{fi} \in \mathscr{F}_{\mathbf{out}}(A_i)$ and w is a test for A_i. But then $\lambda^s(w) = \lambda^s_i(w) \neq \lambda^s_{fi}(w) = \lambda^s_f(w)$. □

Theorem 4.3 *If $A \in \mathbf{Autom}(S)$ and $A_f \in \mathscr{F}_{\mathbf{out}}(A)$, then A_f is testable if and only if A_f is* $\mathbf{bottom}(A)$*-testable. Moreover, a $w \in I^*$ exists, such that w is a test for all testable $A_f \in \mathscr{F}_{\mathbf{out}}(A)$.*

Proof. If A_f is testable, we know a fortiori that A_f is $\mathbf{bottom}(A)$-testable.
Conversely, suppose A_f is $\mathbf{bottom}(A)$-testable. Since $\delta = \delta_f$ $\mathbf{bottom}(A_f)$ is a testable fault for $\mathbf{bottom}(A)$. By le. 4.2 there exists a $w_o \in I^*$ (independent of A_f) such that w_o is a test for $\mathbf{bottom}(A)$ and $\mathbf{bottom}(A_f)$. But w_o is also a test for $\mathbf{bottom}(A)$ and A_f. Let $w = \mathbf{down}(A)w_o$. Using $\delta = \delta_f$, fact 1.6 and the same argument as in the proof of le.3.2 we can show, that A_f ist testable and w is a test for A and A_f. □

Example 4.4 *(testing output faults)* Consider the automata A_k from ex. 2.6 where I = {a, b}. Let $A_f \in \mathscr{F}_{\mathbf{mout}}(A)$ with $\mu_f(0) = 1$. Then A_f is $\mathbf{bottom}(A_k)$-testable and any $w \in I^* \backslash \square$ is a $\mathbf{bottom}(A_k)$-test. By th.4.3 A_f is testable and in connection with ex. 3.1 $a^{2k+1}w$ is a test for A_f, but a^{2k+1} alone ist a test for A_f. If $\mu(0) = 0$, A_f is not testable. If $w \in I^*$ is a test, then $w = xw'$ for $x \in I$ and $w' \in I^*$. If x = a, then the fault given by $\mu_f(2)$ = 1 cannot be tested, and if x = b the fault given by $\mu(1) = 0$ is not testable. Hence, any test for A_f needs at least two testwords. For A_k and $s \in S_k \backslash 0$ we find $\mathbf{evisit}_{A_k}(a^{2k}b^{2k}ab, 1) = (I \times S_k) \backslash (a, 2)$. But then $\mathbf{evisit}_{A_k}$ $(b^i a^{2k} b^{2k} ab, 2i+1)$ = $\mathbf{evisit}_{A_k}(b^{k+i} a^{2k} b^{2k} ab, 2(k-i+1))$ = $(I \times S_k) \backslash (a, 2)$ $(0 \leq i \leq k)$. Now, T_0 = $\{b^i a^{2k} b^{2k} ab | 0 \leq i \leq k\} \cup \{b^{k+i} a^{2k} b^{2k} ab | 0 \leq i \leq k\}$ is a test for every $A_f \in \mathscr{F}_{\mathbf{out}}(A_k)$ with $\lambda_f(a, 2)$ = 0. For $A_f \in \mathscr{F}_{\mathbf{out}}(A_k)$ with $\lambda_f(a, 2)$ = 1 a^{2k+1} is a test. In total $T = T_o \cup a^{2k+1}$ is a test for A_k and $\mathscr{F}_{\mathbf{out}}(A_k)$. □

References

[1] R. BOUTE and E.J. MCCLUSKEY, "Fault Equivalence in Sequential Machines", Technical Report No. 5, Computer Systems Laboratory, Stanford University (June 1971)

[2] W. BRAUER, "Automatentheorie", B.G. Teubner, 1984

[3] E. F. MOORE, Gedanken-Experiments on sequential machines, in: C.E. SHANNON, J. MCCARTHY, Automata Studies, Ann. Math. Studies 34, Princeton University Press, Princeton 1956

[4] J.F. POAGE and E.J. MCCLUSKEY, "Derivation of Optimum Test Sequences for Sequential Machines", Proc. 5th Ann. Sympos. on Switching Theory and Logical Design, pp. 121-132 (1964).