JOURNAL OF ELECTRONIC TESTING: Theory and Applications 14, 305–306 (1999)

Erratum to *An Algebra of Multiple Faults in RAMs*

J.A. BRZOZOWSKI
Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
brzozo@uwaterloo.ca

H. JÜRGENSEN
Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7; and Institut für Informatik, Universität Potsdam, Am Neuen Palais 10, D-14469 Potsdam, Germany
helmut@uwo.ca
helmut@cs.uni-potsdam.de

Received March 22, 1999

Editor: V.D. Agrawal

This note corrects an error in the paper [1], which studies the problem of the represention of multiple faults in RAMs. Our main goal was to characterize multiple faults composed of single faults taken from the Thatte-Abraham fault model [2], that is the fault model consisting of stuck-at, transition, and coupling faults. While our results apply to a much larger class of faults than multiple Thatte-Abraham faults, the class of faults considered in our paper turns out to be too large. The purpose of this note is to explain the error and its correction.

First we give a brief description of the model used in the paper. Suppose that two defects D and D' are present in a RAM, and that automata M and M' represent D and D', respectively. The simultaneous presence of both defects is represented by another automaton constructed from M and M' using an appropriate composition operation.

We model RAMs by component automata. A *component automaton* is a deterministic Mealy automaton $M = (Q, X, Y, \delta, \lambda)$, where Q has the following special property. There exists an integer $n > 0$ and n finite nonempty sets $Q_1, \dots, Q_n$ such that $Q \subseteq Q_1 \times \cdots \times Q_n$. Thus each state $q \in Q$ is an n-tuple $q = (q_1, \dots, q_n)$, where $q_i \in Q_i$ for $i = 1, \dots, n$. A component automaton is called *binary* if $Q_i \subseteq \{0, 1\}$ for all i. We consider only binary component automata.

With the fault-free component automaton M we associate a family of component automata $M' = (Q', X, Y, \delta', \lambda')$, called *delta fault types* of M, which can differ from M in the state set, the transition function, and, trivially, the output function. The state set Q' must be a subset of the state set Q of the fault-free component automaton M. The transition function may be arbitrary. The output function λ' is the restriction of λ to $Q' \times X$.

A set $Q' \subseteq \{0, 1\}^n$ is called a *subproduct* if $Q' = Q'_1 \times \cdots \times Q'_n$ for some sets $Q'_1, \dots, Q'_n \subseteq \{0, 1\}$. A delta fault type is called a *subproduct* fault type if its state set is a subproduct.

Depending on the physical properties of the fault model, two different composition operations $\circ$ and $\diamond$ were introduced. For both operations the state set of the double fault composed of the faults M^1 and M^2 with state sets Q^1 and Q^2, respectively, is the set $Q^1 \cap Q^2$. The transition function of the double fault is defined by Rule 1 for $\diamond$ and Rule 2 for $\circ$. It turns out that there are examples of delta fault types for which these rules are inconsistent with the definition of the state set of the double fault, as we now illustrate.

Example 1. Let M^1 be the stuck-equal fault type, that is, a faulty 2-cell RAM with state set $Q^1 = \{00, 11\}$ and a transition function in which a write to either cell has the same effect on both cells, and the read operation works correctly. Let M^2 be the pattern-sensitive fault type with state set $Q^2 = \{00, 01, 10, 11\}$, in which the write-1 operation to cell 1, w_1^1, fails when the state is 00. All other operations perform correctly. According to Rule 2 defining the $\circ$-composition of automata, the state set of the double fault type $M^{1\circ 2}$ is $Q^{1\circ 2} = Q^1 \cap Q^2 = \{00, 11\}$. Rule 2 then involves a case distinction, depending on whether the state set of the cell being considered is a singleton or not. This is where the basic error occurred. The jth component $Q_j^{1\circ 2}$ of the state set is not defined in [1], but would be naturally defined as the set of all values appearing in cell j. In our example, this set is $\{0, 1\}$ for all j. Hence for $\delta^{1\circ 2}(00, w_1^1)$ only the second case of the rule applies, and $\delta^{1\circ 2}(00, w_1^1) = 01$. However, 01 is not in the state set of $M^{1\circ 2}$.

This problem does not occur if the state set of every fault type considered is a subproduct. Note that the stuck-equal fault type is not a subproduct fault type. The composition operation $\circ$ has already been defined in [3], but only for subproduct automata. Hence the difficulty described above did not appear. In [1] we wanted to generalize the $\circ$-composition to the set of all delta fault types, but this extension failed. To correct the error, it suffices to add the subproduct condition to every statement involving $\circ$ in [1].

There are similar difficulties with the $\diamond$-composition. They are not illustrated by the example above; hence we give a second example.

Example 2. Let M^1 be a faulty 3-cell RAM with state set $Q^1 = \{000, 001, 110\}$. Suppose that $\delta^1(000, w_1^1) = 110$; the other transitions are irrelevant to this discussion and can be arbitrary. Let M^2 have the state set $Q^2 = \{000, 001, 101, 110\}$, let $\delta^2(000, w_1^1) = 101$, and let the other transitions be arbitrary. Rule 1 involves a case distinction, depending on whether the state set of the cell being considered is a singleton or not. The set of all values appearing in cell j is $\{0, 1\}$ for all j. Hence for $\delta^{1\diamond 2}(000, w_1^1)$ only the second case of the rule applies, and $\delta^{1\diamond 2}(000, w_1^1) = 111$. However, 111 is not in the state set of $M^{1\diamond 2}$.

As in the case of $\circ$, the correction to this problem is provided by the addition of the subproduct condition to all statements involving $\diamond$ in [1].

The interested reader is referred to [4] where these issues are further clarified, and where new characterizations of the multiple Thatte-Abraham fault types are given. More specifically, we show the following result:

Theorem 1. (a) *A delta fault type is change-attempt-activated, pattern-insensitive, and a subproduct fault type if and only if it is equivalent to a $\circ$-composition of Thatte-Abraham fault types.* (b) *A delta fault type is change-success-activated, pattern-insensitive, and a subproduct fault type if and only if it is equivalent to a $\diamond$-composition of Thatte-Abraham fault types.*

References

1. J.A. Brzozowski and H. Jürgensen, “An Algebra of Multiple Faults in RAMs,” *J. of Electronic Testing: Theory and Applications*, Vol. 8, 129–142, 1996.
2. S.M. Thatte and J.A. Abraham, “Testing of Semiconductor Random-Access Memories,” *Digest of Papers, 7th Int. Conf. on Fault-Tolerant Computing*, Los Angeles, June 28–30, 1977, pp. 81–87.
3. J.A. Brzozowski and H. Jürgensen, “Component Automata and RAM Faults,” *Proc. 2nd International Colloquium on Words, Languages, and Combinatorics, Kyoto, Japan, 1992*, M. Ito and H. Jürgensen (Eds.), World Scientific, Singapore, 1994, pp. 49–67.
4. J.A. Brzozowski and H. Jürgensen, “Semilattices of Fault Semiautomata,” In *Jewels are Forever*, J. Karhumäki, H. Maurer, G. Păun, and G. Rozenberg (Eds.), Springer-Verlag, Berlin 1999, pp. 3–15.