is defined in terms of the predicate calculus and involves formulas without quantifiers.

Let $TM$, $LBA$, $MTA_2$ stand for Turing machines, linear-bounded automata, and two-way multitape finite automata, respectively. Let $TR_0$, $TR_1$, $TR_d$ denote classes of predicate formulas involving transitive closure. The subscript $d$ stands for deterministic. The main results in the paper are the six equalities in the diagram. The proofs are mainly simulations.

$$
\begin{align*}
TM & = TR_0 & TM_d & = TR_d \\
& \cup & \cup & \\
LBA & = TR_1 & LBA_d & = TR_d \\
& \cup & \cup & \\
MTA_2 & = TR_2 & MTA_d & = TR_d
\end{align*}
$$

The author believes that this characterization in terms of formulas will provide much needed new tools for the description and analysis of languages accepted by various automata. Some previous results about the relation between languages accepted by automata and recursive functions are extended.

D. TISCHRITZIS
Dept. of Computer Sci.
University of Toronto
Toronto, Canada


A stationary maximin automaton is a system $A = (\langle S, U, f, h, F \rangle, \langle s, u, s' \rangle)$, where $S$ and $U$ are finite nonempty sets of states and inputs, respectively; $F \subseteq S$ is a set of final states; $f : S \times U \times S \rightarrow \{0, 1\}$ can be imagined to be a transition relation which associates with every state $s \in S$ and input $u \in U$ a measure, $0 \leq f(s, u, s') \leq 1$ that the next state is $s'$, there being no constraint, for example, that the sum of these measures equals 1 for a given $s$ and $u$. Similarly, the function $h : S \rightarrow \{0, 1\}$ defines something like an initial distribution, again with no constraints on the sum of all of the $h(s)$'s. What serves to distinguish maximin automata from other better known classes of automata is the manner in which the function $f$ is extended to sequences of inputs $x \in U^*$; this is defined as follows:

$$f^*(s, xu, s') = \max_{s \in S} \{ f^*(s, x, s') \}.$$ 

If one studies this definition of maximin automata and, in particular, the definition above of $f^*$, one can see that maximin automata are nothing more than weighted, directed graphs, or networks, and $f^*$ is strikingly similar to the concept of a flow in such a network [1].

Having defined maximin automata, and pointed out their generality, i.e., they contain both deterministic and nondeterministic automata as special cases, Santos devotes the majority of the paper to showing that “most of the basic concepts [and results] of existing automata, e.g., equivalences, reduction, behavior, etc., may be carried over to maximin automata with appropriate modifications.” Indeed, they can and quite simply; none of the proofs requires more than a few sentences, and none of them departs very much at all from the corresponding results for deterministic automata.

Stated in other words, this paper is basically a straightforward reworking of parts of the classic papers in automata theory contained in [2] through [4].

Unfortunately, the first two of these three papers receive no mention in the paper; in addition, four of the nine papers cited as references receive no mention in the paper.


The author uses the Muller–Bartky asynchronous circuit theory to study the problem of designing circuits whose physical implementations operate correctly despite unpredictable signal delays in the wires connecting the elements.

To achieve this end, the author defines restrictions and extensions of states, sequences of states, and circuits. A good extension $C^*$ is said to be one whose observable behavior, at the nodes corresponding to those of $C$, is identical to that of $C$. Specific types of extensions are next defined and studied. The delay problem of the first kind occurs when some circuit $C^{**}$, obtained from $C$ by replacing wires of $C$ by sequences of delay elements, is a bad (not good) extension of $C$.

In the next section (Section 6) the $m(h)$ and $\mu(h)$ functions and the memory index $\mu$ of one finite sequence, $\{a(h)\}$ of signal values, with respect to another, $\{b(h)\}$, were defined and examined. With these tools, the author in Section 7 proves his principal result. Let $C$ be a circuit in which each delay wire is fed by an ordinary node having two states, and at most one delay wire feeds any ordinary node. Then $C$ suffers from the delay problem of the first kind if and only if an extension $C^*$ of $C$, formed by replacing each delay wire of $C$ by a series connection of three delay elements, is a bad extension.

The delay problem for asynchronous circuits was introduced by this reviewer for a particular subcase of the general Muller–Bartky theory. The present paper, in extending these results, is also attempting to attack theoretically similar problems. First, physical realizations of switching circuits operate on a continuum of values by using switching thresholds to simulate discrete levels; different physical circuits have different signal thresholds. Second, designers may wish to synthesize complex logical functions from primitives, while preserving the idealized operational characteristics.

The paper was quite interesting to read, and provides a good example of the power of the Muller–Bartky theory. However, Sections 6 and 7, concerning the memory index and its application to the principal theorem, were quite difficult to follow. In fact, the proof of Lemma 7.4 is incorrect as it stands. The author’s usage of the $\delta$ function as the binary complement of the Kronecker $\delta$ function may trap the unwary reader. Six typographical errors were noted, but these should not seriously disturb the student.

The paper is a valuable contribution to the theory of asynchronous circuits. This reviewer is eagerly awaiting the appearance of Part II.

M. H. SHELLY
Poughkeepsie, N. Y.


Applying some simple, easily understood principles, Spira, in extending some earlier work of Winograd, points the way to a powerful theory of computation complexity. Spira considers a $(d, r)$ combinational network

1 The reader is also referred to pp. 75–80 of Arbib’s recent book [1] for an abridgment of this paper.
which is an interconnection of $r$-input, single-output modules, with each input-output line carrying a value from the set $\{0, 1, \ldots, d-1\}$. A finite function $\phi: X_1 \times X_2 \times \cdots \times X_n \rightarrow Y$ is to be computed, but it is assumed that before the inputs are inserted into the network, each input can be individually (and arbitrarily) transformed by a set of maps $g_i: X_i \rightarrow I_i$. It is also assumed that there is a 1-input map $h: Y \rightarrow O_1$, so in practice the $(d, r)$ network will have as input $[g_1(x_1), \ldots, g_n(x_n)]$ and as output $h(\phi(x_1, \ldots, x_n))$.

The problem is to bound the number of levels required of the network. Given a $\phi$ for a particular output mapping, it is not difficult to specify a lower bound on the number of groups required, by identifying for each output line the number of different values of input variables which yield a different output value. The minimum number of levels required for each output line is then evaluated by noting that an output at level $t$ can depend on at most $r^t$ input lines whence the output line requiring the most levels provides the bound. It is evident that the bound is critically dependent upon the output mapping, but for most functions the relationship between the bound and the mapping has not been identified.

One notable exception is the case of multiplication in a finite group $G$, in which case the function is $\phi: G \times G \rightarrow G$. Spira shows that the bound in this case is simply related to the quantity $B(G) = \min_{e \in G} [\text{maximum order of any subgroup of } G \text{ not containing } e]$. Surprisingly, a network (albeit quite costly) can be specified which requires a number of levels within one of the lower bounds. Assume for $G$ a set of subgroups $K_1, \ldots, K_n$ where $\bigcap_{i=1}^n K_i = \{e\}$, $e = \text{identity}$, and such that $\max_j [K_j]$ is minimized over all such sets. Since each element of $G$ is uniquely specified by identifying in which right coset of each $K_i$ it is contained, the network in computing $ab$, $a, b \in G$ need “merely” determine for each right coset of each $K_i$ if $ab$ is contained in that coset. The set of output lines of the network then corresponds to the aggregate of these binary decisions and a simple construction is given to compute this decision.

One need only develop the network for a particular group according to the rules in order to appreciate the extraordinary number of modules required, and also to note that in many cases an “obvious” implementation will achieve the bound. In the case of $G = C_2 \times C_2 \times C_2$, where $C_2$ is the cyclic group of order 2, assuming $d = r = 2$, Spira’s construction requires six modules (two mod 2 adders for each of three subgroups) in addition to an encoder and decoder of reasonable complexity, while the bound (of 1 level) is easily achieved with three mod 2 adders by treating each element of $G$ as an element of a three-dimensional vector space over $GF(2)$.

In fairness to Spira, most groups do not admit to such obvious implementations as the above group.

There remains the interesting theoretical problem of identifying for various groups a minimum number of cosets such that each group element is uniquely identified; this would permit less costly realizations. Perhaps of more importance is the development of complexity measures which reflect, in addition to the number of levels, such factors as number of modules, number of module types, and the cost of encoding and decoding. It also appears fruitful to apply the general theory to other functions besides group multiplication as Savage [2] has done for the case of decoding BCH multiple error correcting codes. Despite the present practical shortcomings of the technique for implementing group multiplication, I believe Spira’s (and Winograd’s) results are an embarkation towards a rich theory of computation complexity.

KARL N. LEVITT
Stanford Research Institute
Menlo Park, Calif.

REFERENCES