On the Power of Bounded Concurrency I: Finite Automata

DORON DRUSINSKY AND DAVID HAREL

The Weizmann Institute of Science, Rehovot, Israel

Abstract. We investigate the descriptive succinctness of three fundamental notions for modeling concurrency: nondeterminism and pure parallelism, the two facets of alternation, and *bounded cooperative concurrency*, whereby a system configuration consists of a bounded number of cooperating states. Our results are couched in the general framework of finite-state automata, but hold for appropriate versions of most concurrent models of computation, such as Petri nets, statecharts or finite-state versions of concurrent programming languages. We exhibit exhaustive sets of upper and lower bounds on the relative succinctness of these features over Σ^* and Σ^ω, establishing that:

(1) Each of the three features represents an exponential saving in succinctness of the representation, in a manner that is independent of the other two and additive with respect to them.
(2) Of the three, bounded concurrency is the strongest, representing a similar exponential saving even when substituted for each of the others.

For example, we prove exponential upper and lower bounds on the simulation of deterministic concurrent automata by AFAs, and triple-exponential bounds on the simulation of alternating concurrent automata by DFAs.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*automata*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*alternation and nondeterminism, parallelism and concurrency*

General Terms: Languages, Theory

Additional Key Words and Phrases: Alternation, bounded cooperative concurrency, finite automata, nondeterminism, omega-automata, statecharts, succinctness

1. *Introduction*

Numerous models have been proposed for capturing the parallelism inherent in real-world concurrent systems. Direct communication and shared memory models are the main general approaches. Much research has been carried out

A preliminary version of the paper appeared as: "On the power of cooperative concurrency." In *Proceedings of Concurrency '88.* Lecture Notes in Computer Science, vol. 335. Springer-Verlag, New York, 1988, pp. 74–103.

D. Harel holds the William Sussman Chair in Mathematics. His research was partially supported by a grant from the Gutwirth Foundation.

Authors' current addresses: D. Drusinsky, 20654 Gardenside Circle, Cupertino, CA 95014; D. Harel, Dept. of Applied Mathematics and Computer Science, The Weizmann Institute of Science, 76100 Rehovot, Israel.

on the relative merits of these, on their semantics, implementability, naturalness, etc. As far as assessing the fundamental savings that these models offer in the face of sequential models or in the face of each other, most of the work has been done in the standard framework of complexity theory, namely by evaluating the time, space and number of processors required to solve various problems, and by searching for trade-offs between these measures. However, in distributed and concurrent computing, it is not only computation with general-purpose resource-bounded mechanisms that is of interest, but also, and increasingly more so in recent years, the medium of *finite-state* protocols and programs. In these, *terminating* computations are modeled by finite words over a finite alphabet Σ, and *ongoing* computations are modeled by infinite words thereof. The basic mechanism for programming and generating such computations is that of finite automata (or ω-automata in the latter case). It is reasonable, therefore, to attempt a thorough investigation of the relative power of features suggested for modeling concurrency, within the basic framework of finite automata.

What features should we be looking at? Existential and universal branching are perhaps the most popular ways of modeling parallelism in complexity theory.[1] However, unlike the constructs used in the study of real systems, in these types of branching no communication takes place between the spawned processes, except when time comes to decide whether the input should be accepted. In Turing machines, for example, this fact manifests itself in the totally separate tapes that are assumed to be generated whenever branching (of either kind) takes place. It would appear that in order to capture real-world concurrency we would want to allow the mechanism, during a single computation, to be possibly in more than one state, and these states to be able to cooperate in achieving a common goal. This approach, which we might call *cooperative concurrency*, is the dominating one in the research community of concurrent and distributed systems, and not the noncooperative concurrency featured in the alternation approach. Moreover, in the real world the number of processes available for simultaneous work is bounded and cannot be assumed to grow as the size of the input grows. In contrast, existential and universal branching are unbounded: New processes can be spawned without limit as the computation proceeds (i.e., as the length of the input word grows). The motivation for this paper (and its companions [Globerman and Harel 1994; Harel 1989; Harel et al. 1990; Hirst 1989; and Hirst and Harel 1994]) is to determine how bounded cooperative concurrency, or simply *bounded concurrency* for short, fares with respect to the two classical kinds of branching, in the realm of finite automata and their extensions.

What should the criteria for comparing such features be? Pure power of expression is irrelevant here, since all reasonable variants of finite automata—including the ones we introduce below—accept the regular sets over Σ^* and (for the acceptance criteria we use here) the ω-regular sets over Σ^ω.[2] Time and space, in the usual complexity-theoretic sense, are not relevant either, since finite automata operate in real-time and, apart from the states themselves, they have no additional storage. The correct measure, therefore,

[1] Typically, the adjectives *alternating*, *nondeterministic*, and *deterministic* are used to denote the presence of both of these, the presence of the first and the absence of both, respectively.

[2] However, see the discussion on synchronized automata below.

seems to be *succinctness*, that is, the inherent *size* of an automaton required to accept a given language.

This paper is concerned with seeking exponential (or higher) discrepancies in the succinctness of finite automata when augmented with the various mechanisms for modeling concurrency, over finite and infinite computations alike.

Nondeterminism and pure and-parallelism are well understood in automata, and take the form of the ∃-states and ∀-states in the alternating variant of finite automata (AFAs; see Chandra and Stockmeyer [1976], Chandra et al. [1981], and Kozen [1976]). Indeed, regarding succinctness, it is well-known that NFAs are exponentially more succinct than DFAs, in the following upper and lower bound senses (see Meyer and Fischer [1971], Rabin and Scott [1959]).

—Any NFA can be simulated by a DFA with at most an exponential growth in size.
—There is a family of regular sets, L_n, for $n > 0$, such that each L_n is accepted by an NFA of size $O(n)$ but the smallest DFA accepting it is at least of size 2^n.

By duality, the same is true of what are sometimes called ∀-automata, namely, the dual machines, in which all branching is universal. It is also true that AFAs, that is, those that combine *both* types of branching, are exponentially more succinct than either NFAs or ∀-automata, and indeed are *double*-exponentially more succinct than DFAs (see Chandra and Stockmeyer [1976], Chandra et al. [1981], and Kozen [1976]). These results also hold in both the upper and lower bound senses described,[3] so that if we denote nondeterminism by E and parallelism by A, these known results can be summarized as in Figure 1.[4] These results thus establish that, in the framework of finite automata, E and A are exponentially powerful features, independently of each other (i.e., whether or not the other is present), and, moreover, their power is additive: the two combined are double-exponentially more succinct than none.

In order to model bounded concurrency, which we shall denote by C, we could have chosen bounded, finite-state versions of any one of a large number of proposed models of computation, such as Petri nets [Reisig, 1985], CSP [Hoare, 1978], CCS [Milner, 1980], statecharts [Harel, 1987], or of any of the accepted concurrent programming languages. We have decided, however, to remain as close as possible to classical finite automata, and thus propose a simple version of cooperating finite automata. Of all the aforementioned models, it comes closest to the language of statecharts, which we shall use to illustrate some of our constructions. In fact, cooperating automata correspond to statecharts consisting of a single collection of orthogonal components [Harel, 1987], each of which is merely a finite automaton. Nevertheless, our results are very robust, and hold for the bounded, finite-state variants of virtually all other models (e.g., Petri nets with a bounded number of tokens, or CSP and CCS with finitely many states and bounded depth of recursion), since

[3] Of course, the double-exponential lower bound does not follow trivially from the single-exponential bounds for the two separate features, since different examples may have been used for each of them.

[4] By convention, solid lines are assumed to represent one-exponential upper and lower bounds, and additive transitivity is assumed too, so that the line labeled "two-exponentials" that would lead from (E, A) to ∅ is omitted for clarity, despite the fact that it does not follow *a priori*.

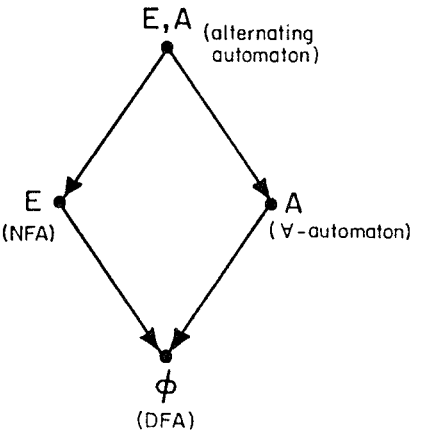

FIG. 1. Known bounds for classical automata.

there are rather straightforward polynomial reductions between any two of them.[5]

Our first set of results establishes the solid lines of Figure 2 and all the transitivity consequences thereof. Among other things, these include exponential upper and lower bounds for simulating nondeterministic concurrent machines (e.g., nondeterministic bounded-token Petri nets, or nondeterministic statecharts) on NFAs, double-exponential bounds for simulating them on DFAs, and, when ∀-states are added, a *triple*-exponential bound for simulating *alternating* concurrent machines on DFAs. The solid lines of Figure 2 thus show that bounded concurrency (C) represents a third, separate, exponentially powerful feature; it is *independent* of conventional nondeterminism (E) and parallelism (A), since the savings remain intact in the face of any combination of A and E; and it is also *additive* with respect to them, by virtue of the double- and triple-exponential bounds along the appropriate compound lines in the figure. This result is of interest, as it shows, among other things, that the unbounded nature of the pure AND of alternation prevents it from being subsumed by the bounded AND of the C feature, and the cooperative nature of the AND in the C feature (as embodied by the joint transitions in Petri nets or statecharts, for example) prevents it from being subsumed by the noncooperative AND of alternation.

Our next set of results considers the more delicate question of how C compares with A and E themselves using the same yardstick, namely, possible exponential discrepancies in succinctness. For example, the results just described do not say anything about the possibility of an exponential gap between E and C. Here our results for the finite word case are summarized by the four dashed lines and the one doubly dashed line in Figure 2. Each of the singly dashed lines denotes exponential upper and lower bounds for the simulation in the downward direction and polynomial bounds for the upper direction. In

[5]Clearly, these bounded, finite-state versions would have to be defined carefully, especially when nondeterministic or alternating variants are considered. An alternating Petri net, for example, would allow branching transitions, so that a transition that is fired can take one of several possibilities, and branches or places are labeled as existential or universal. The main point, however, is that our proofs use these features in very simple ways (e.g., concurrency to count in binary, and nondeterminism to choose an arbitrary appearance of some symbol in the input word), so that the reader should have little problem with our general claim of robustness.

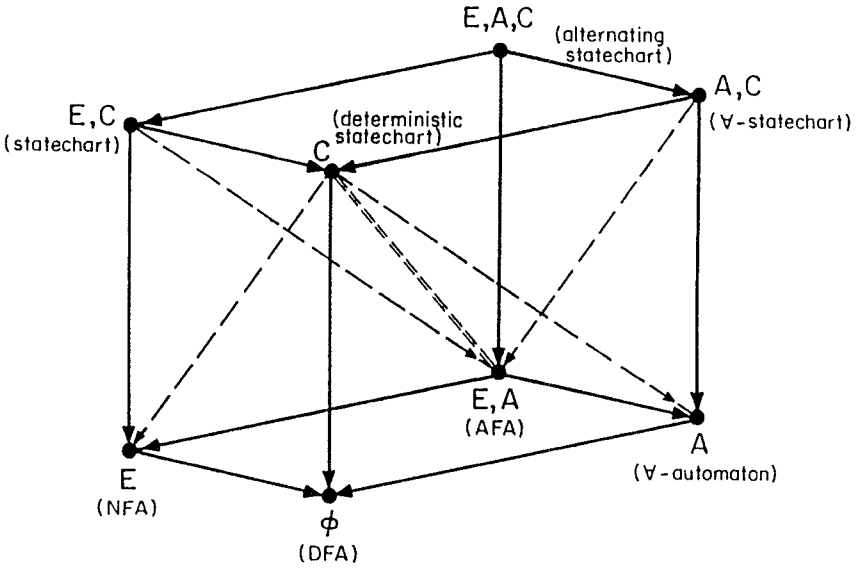

FIG. 2. Summary of results for the Σ^* case.

particular, the nondeterministic statecharts of Harel [1987] are shown to be exponentially more succinct than AFAs, and the same holds when nondeterminism is absent from both. The doubly dashed line represents upper and lower exponential bounds in *both* directions, meaning that alternation and bounded concurrency can be simulated by each other with at most an exponential growth in size, and that, in general, neither exponential gap can be eliminated. We consider these to be our most interesting results, as they show that bounded concurrency is actually *more* powerful than each of pure parallelism or nondeterminism taken alone, and is comparable in power to the combination of both (by virtue, ironically, of the fact that the two are technically incomparable). The results for finite words appear in Sections 3 and 4.

For the infinite word case, things are somewhat more complicated, since there are a variety of acceptance criteria that differ in subtle ways. Our results, which are almost exactly the same as in the finite word case, pertain to the Rabin and Streett criteria, and are illustrated in Figure 3. As the figure shows, vertical lines and lower horizontal lines apply to both Rabin and Streett criteria; upper horizontal lines on the left-hand side of the figure use the Rabin criterion and those on the right-hand side use the Streett criterion. We actually conjecture that the entire cube holds for each of the criteria separately. Some of the upper bounds require using (and extending) state-of-the-art results on the determinization of ω-automata. For example, we base the polynomial transition from nondeterministic Rabin automata to deterministic Rabin C-machines on Safra's exponential determinization construction [Safra, 1988]. The results for infinite words appear in Sections 5 and 6.

This is one in a series of papers regarding the inherent power of bounded cooperative concurrency and its relationship to nondeterminism and parallelism. In Hirst and Harel [1994], we deal with the context-free case, via pushdown automata enriched with the E, A, and C features (see also Hirst

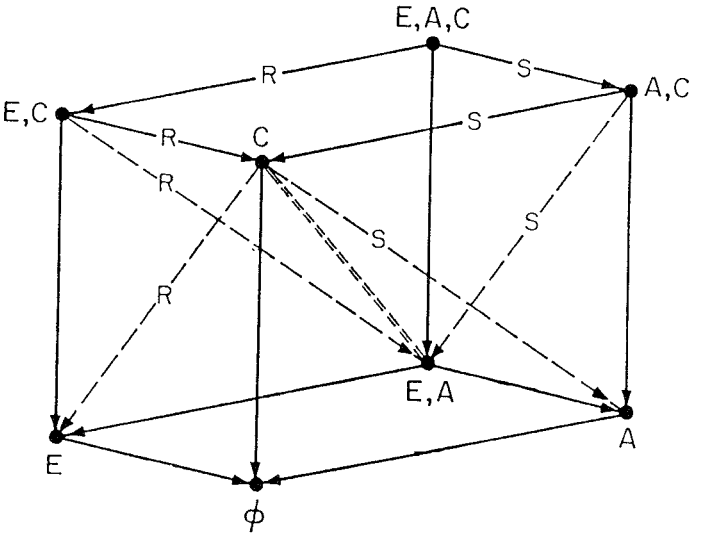

FIG. 3. Summary of results for the Σ^{ω} case.

[1989]), in Globerman and Harel [1994], we consider the framework of pebble automata, and in Globerman and Harel [1994]; and Harel et al. [1990], we investigate the effect these features have on the complexity of reasoning about propositional concurrent programs. Hirst [1989] also contains results on finite automata over one-letter alphabets. A noteworthy fact that surfaces repeatedly, is that the C feature retains its inherent exponential power in virtually all of the cases we have looked at, whether the other features turn out to be more powerful or less so. Many of the results in these papers are surveyed in Harel [1989].

To complete the background, we should mention *synchronized automata*, an idea that was first published in Slobodová [1988] and which has been the subject of extensive research since.[6] A synchronized automaton is one in which universal branching, that is, the A feature in our terminology, is enriched with "synchronizing states," which enable the parallel branches to cooperate. As is the case here, it appears that the precise form of cooperation is unimportant and that their results are similarly robust. The main difference, however, is that the cooperative concurrency in synchronized automata is unbounded, since it is the parallel processes of the A feature that are used as carriers for the cooperative work. As shown in several of the papers on synchronized automata, this ability takes finite automata out of the realm of regular sets. In fact, in Hromkovič et al. [1992], the following interesting result is proved: one-way synchronized alternating automata (which are really (E, A)-machines, but with cooperation allowed within the A feature) accept exactly the context-sensitive sets.

Thus, the synchronized automata approach enriches an already existing feature of automata, in a way that extends the class of languages accepted.

[6] This work and ours were independent; the conference version of the present paper was also published in 1988 [Drusinsky and Harel, 1988].

Research then centers on the added *expressive power* of the resulting machines. In contrast, our approach adds a new, independent, feature to finite automata, but one that does not introduce any new languages. Research then centers on the added *succinctness* of the resulting machines.

2. *Definitions*

In this section, we define automata augmented with the E, A, and C features, and call them (E, A, C)-AUT, or sometimes just (E, A, C)-*machines*. As special cases, if the A and C features are not present, the resulting E-AUT are simply NFAs. Similarly, (E, A)-AUT are AFAs, and ∅-AUT are ones with none of the three features, and hence are simply DFAs.

Let Σ be a finite alphabet. An (E, A, C)-AUT M is a tuple

$$M = (M_1 \cdots M_v, \Phi, \Psi)$$

for some $v \geq 1$, where each M_i is a triple (Q_i, q_i^0, δ_i). Here, Q_i is a finite set of states (the Q_i, for $1 \leq i \leq v$, are required to be pairwise-disjoint), $q_i^0 \in Q_i$ is the initial state, and δ_i, the transition table, is a finite subset of the product $Q_i \times \Sigma \times \Gamma \times Q_i$.[7] We use Γ to denote the collection of propositional formulas over the alphabet of the atomic letters $\cup_{1 \leq j \leq v} Q_j$. Finally, Φ, the E-*condition*, and Ψ, the *termination condition*, are elements of Γ.

The intuition is that M consists of v automata (sometimes called M's *orthogonal components*, or simply *components* for short), each with its own set of states, initial state and transition table. These automata work together in a synchronous manner, taking transitions according to the (common) input symbol being read, their internal states, and the *condition formulas* from Γ. These are interpreted to take on truth values according to the states of possibly all the v components. Φ distinguishes between existential and universal state configurations (i.e., between E and A states), and Ψ indicates halting configurations.

More formally, a *configuration* of M is an element of $Q_1 \times Q_2 \times \cdots \times Q_v \times \Sigma^* \times \mathcal{N}$, indicating the state each of the M_i is in, the input word and the position of M in the word. Clearly, $m \leq |x|$ must hold for any configuration $(q_1, \ldots, q_v, x, m)$. We say that a configuration c *satisfies* a condition $\gamma \in \Gamma$, if γ evaluates to *true* when each symbol therein is assigned *true* iff it appears in c. Thus, for example, the condition $(q \vee p) \wedge \sim r$, where $q, p \in Q_1$ and $r \in Q_2$, will be satisfied by any configuration for which M_1 is in state q or p, and M_2 is not in state r.

To define the behavior of M, let $x = x_1 x_2 \cdots x_k$ be a finite word over Σ, and let $t = (q, a, \gamma, p)$ be a transition in M_i's transition table δ_i. We say that t is *applicable to* a configuration $c = (q_1, \ldots, q_v, x, j)$, if $x_j = a$, $q_i = q$, and c satisfies γ. A configuration $(p_1, \ldots, p_v, x, m)$ is said to be a *successor of* c if for each i there is a transition $(q_i, x_j, \gamma_i, p_i) \in \delta_i$ that is applicable to c, and $m = j + 1$.

A configuration is *existential* if it satisfies the E-condition Φ, otherwise it is *universal*. It is *accepting* iff it satisfies the termination condition Ψ.

[7] The definitions could have been given to include ϵ-moves too, by taking δ_i to be a finite subset of $Q_i \times (\Sigma \cup \epsilon) \times \Gamma \times Q_i$, and modifying the other parts of the definitions accordingly. Our results all hold for this version too.

A *computation* of M on $x \in \Sigma^*$ is defined in a way very similar to that of AFAs [Chandra and Stockmeyer, 1976; Chandra et al., 1981; Kozen, 1976]. It consists of a tree, each node of which is labeled with a configuration. The root is labeled with the *initial* configuration $(q_1^0, q_2^0, \ldots, q_v^0, x, 1)$, and a node has one successor node for each of its label's successor configurations, labeled with that successor configuration. Nodes are assigned 1/0 (accept/reject) marks, in a bottom up manner, as in the definition for AFAs, ORing the marks of the successors of an existential node and ANDing those of a universal node. The input word x is accepted iff the root gets marked 1.

The *size* of the machine $M = (M_1 \cdots M_v, \Phi, \Psi)$ is

$$|M| = |\Phi| + |\Psi| + \sum_{i=1}^{v} |M_i|,$$

where the size of a formula in Γ is simply its length in symbols, and the size of each component automaton is defined by

$$|M_i| = |Q_i| + \sum_{(q,a,\gamma,p) \in \delta_i} (3 + |\gamma|).$$

Note that, as special cases, if $v = 1$, the machine M is simply an alternating finite automaton, that is, an AFA (in our terminology, it is an (E, A)-AUT); if Φ is *true*, then all states are existential, so that M is an NFA (an E-AUT); if Φ is *false*, then all states are universal, so that M is an $\forall$-automaton (an A-AUT); if each configuration has at most one successor, then M is deterministic, that is, it is a C-AUT.

In some of the proofs later, we illustrate the construction of various machines using statecharts [Harel, 1987], which can be viewed as employing the C feature in a more flexible way. In general, statecharts can be made to conform to the terms of the above definitions with at most a linear increase in size. Although we shall not prove this general claim here, the reader will be able to apply it easily to the examples we use.

We extend the definitions of (E, A, C)-AUT to define acceptance over Σ^ω, and call the resulting machines (E, A, C)-ωAUT. The termination condition is enriched, so that, rather than a single formule Ψ from Γ, we have a *finite* set of pairs of conditions

$$\Omega \subseteq (\Gamma \times \Gamma).$$

A *run* over a word $x \in \Sigma^\omega$, is an infinite sequence r of successive configurations, and $inf(r)$ is the set of configurations appearing in r infinitely often. We shall concentrate on two acceptance criteria, Rabin's [Choueka, 1974; Rabin, 1969] and Streett's [1982]. First, assume for the moment that the machine M is deterministic (and total), so that there is exactly one run per input word x; call it r_x. The two acceptance criteria are now defined as follows:

(1) *Rabin acceptance.* The machine M R-*accepts* a word $x \in \Sigma^\omega$, if there is a pair $(\Psi_1, \Psi_2) \in \Omega$, such that there is a configuration in $inf(r_x)$ that satisfies Ψ_1, but no configuration therein satisfies Ψ_2.

(2) *Streett acceptance.* The machine M S-*accepts* a word $x \in \Sigma^\omega$, if for each pair $(\Psi_1, \Psi_2) \in \Omega$, if there is a configuration in $inf(r_x)$ that satisfies Ψ_1, then there is also a configuration therein that satisfies Ψ_2.

The conventional Rabin and Streett criteria for DFAs can be easily seen to be a special case of this definition. Simply take $v = 1$ and choose the pairs of conditions (Ψ_1, Ψ_2) as conjunctions that specify the pairs of sets of states required for these classical criteria.

Extending the definition to (E, C)-ωAUT is easy: M accepts x if there is at least one run on x that accepts. For (A, C)-ωAUT, all runs on x must accept. For the general case of alternation, acceptance is, again, a straightforward adaption of the usual definition for AFA. Since the computation tree is infinite, it is inconvenient to talk about marking it with 0's and 1's; it is easier here to consider traces. A *trace* of an (E, A, C)-ωAUT on a word $x \in \Sigma^\omega$, is a subtree of M's computation tree on x, that includes *all* offspring of each universal node and one offspring of each existential node. M accepts x if there is a trace of M on x, for which every path adheres to the appropriate acceptance criterion.

We say that two (E, A, C)-ωAUT are R-*equivalent* (respectively, S-*equivalent*) if they R-accept (respectively, S-accept) the same subset of Σ^ω.

We now define the exponential and multi-exponential gaps we are interested in establishing between the various kinds of machines. Let ξ be any subset of {E, A, C}. We denote by ξ-AUT and ξ-ωAUT the classes of machines employing the features in ξ.

Definition. Let ξ_1 and ξ_2 be any two subsets of {E, A, C}. We shall write $\xi_1 \overset{p}{\to} \xi_2$ (respectively, $\xi_1 \dot{\to} \xi_2$, $\xi_1 \ddot{\to} \xi_2$, or $\xi_1 \dddot{\to} \xi_2$), if there is a polynomial p and a constant $k > 1$, such that, for any machine $M_1 \in \xi_1$-AUT of size n there is an equivalent $M_2 \in \xi_2$-AUT of size no more than $p(n)$ (respectively, $k^{p(n)}$, $k^{k^{p(n)}}$, or $k^{k^{k^{p(n)}}}$).

Definition. Let ξ_1 and ξ_2 be any two subsets of {E, A, C}. We shall write $\xi_1 \overset{\cdot}{\to} \xi_2$ (respectively, $\xi_1 \overset{\cdot\cdot}{\to} \xi_2$, or $\xi_1 \overset{\cdot\cdot\cdot}{\to} \xi_2$) if there is a family of regular languages L_n, for $n > 0$, a polynomial p and a constant $k > 1$, such that L_n is accepted by a machine $M_1 \in \xi_1$-AUT of size $p(f(n))$ for some monotonically-increasing function f, but the smallest $M_2 \in \xi_2$ accepting it is at least of size $k^{f(n)}$ (respectively, $k^{k^{f(n)}}$, or $k^{k^{k^{f(n)}}}$).

When a small R or S is added as a subscript to the arrows in these definitions, they are to be considered as applying to ξ-ωAUT, rather than to ξ-AUT, and to denote R-equivalence or S-equivalence, respectively.

3. *Upper Bounds for the Σ^* Case*

We first establish exponential upper bounds for the vertical arrows of Figure 2. (Among other things, this shows, of course, that concurrent automata accept only the regular languages.)

PROPOSITION 1. *Let ξ be any subset of* {E, A}. *Then* $(\xi, \mathrm{C}) \dot{\to} \xi$.

PROOF. We have to show how to remove the C feature with at most an exponential increase in size. The idea is simply to simulate the behavior of a (ξ, C)-AUT M by a ξ-AUT whose set of states is the Cartesian product of the states in M's component machines, $M_1, \ldots, M_v$. The transition predicates of the simulating machine are written explicitly in terms of the states of the M_i, so that there cannot be more transitions than elements of the Cartesian

product. Clearly, if M is deterministic, nondeterministic, or alternating, the resulting machine will be of the corresponding type too. □

Next, we establish polynomial upper bounds for the upward direction of the four dashed diagonal lines of Figure 2, so that, for example, nondeterminism can be replaced by bounded concurrency with only a polynomial increase in size. This is true whether or not A is present too.

PROPOSITION 2. *Let* ξ *be* $\varnothing$ *or* A. *Then* $(\xi, \mathrm{E}) \xrightarrow{\mathrm{p}} (\xi, \mathrm{C})$. *The same holds with* A *and* E *exchanged.*

PROOF. For the simulation of E by C we mimic the classical subset construction for eliminating nondeterminism [Rabin and Scott, 1959], using a collection of orthogonal components, one for each of the n states in the original NFA. For any state p therein, the corresponding component has two states, indicating, respectively, whether we are in p or not. Subsets of the original state set are represented in the obvious way by these yes/no combinations. When a is read as an input, and p's component is in its no-state, it moves into its yes-state if any of p's a-predecessors in the original NFA is in *its* yes-state right now. Dually, if p is in its yes-state, it moves to its no-state when a arrives, if all of its a-predecessors in the original NFA are in their no-states.

Clearly, the machine accepts if and only if at least one of the components that represent a final state of the NFA is in its yes-state. Also, if the NFA had m transitions, the formulas describing the transitions in the resulting C-machine can be constructed so that their total size is no more than $(m + n)^2$.

The simulation of an A-machine by a C-machine is identical, except for the acceptance decision, which is made only if *all* the components representing final states of the NFA are in their yes-states.

Replacing E by C in the presence of A, that is, simulating an (E, A)-machine by an (A, C)-machine, is a little more subtle. Figure 4 illustrates the construction as applied to a simple AFA, and, for convenience, the simulating machine has been depicted as a ∀-statechart. The basic idea is as above, namely, to mimic the subset construction using components with yes/no-states. However, we now have and-branchings in the original machine, that have to be maintained in the simulation. We do this by incorporating all such branchings one step ahead of time, at the moment a universal state is entered.

Here is a more rigorous description of the construction. Let M be the original (E, A)-AUT, whose state set Q is of size n. Also, let the size of the alphabet be m. The simulating (A, C)-AUT N has n orthogonal components, one for each $q \in Q$. Since M has but one component, its E-condition Φ can be viewed as simply specifying that some of the states in Q are existential and some are universal. The component (in N) of an *existential* state $q \in Q$ contains the usual yes- and no-states; call them q and $\hat{q}$, respectively. However, if q is *universal* in M, the corresponding component in N will have several yes-states, corresponding to the various combinations of possibilities it may carry out when the next symbol is read. Specifically, besides the no-state $\hat{q}$, the component will have a yes-state $q^{q_1, \ldots, q_m}$ for each collection of and-choices available to state q in M in response to the arrival of any of the m letters of the alphabet Σ. For example, if $\Sigma = \{a, b, c\}$, and if the possible transitions in M from state q are (q, a, p), (q, a, r), (q, b, q), (q, b, s), (q, b, r), and $(q, c,$

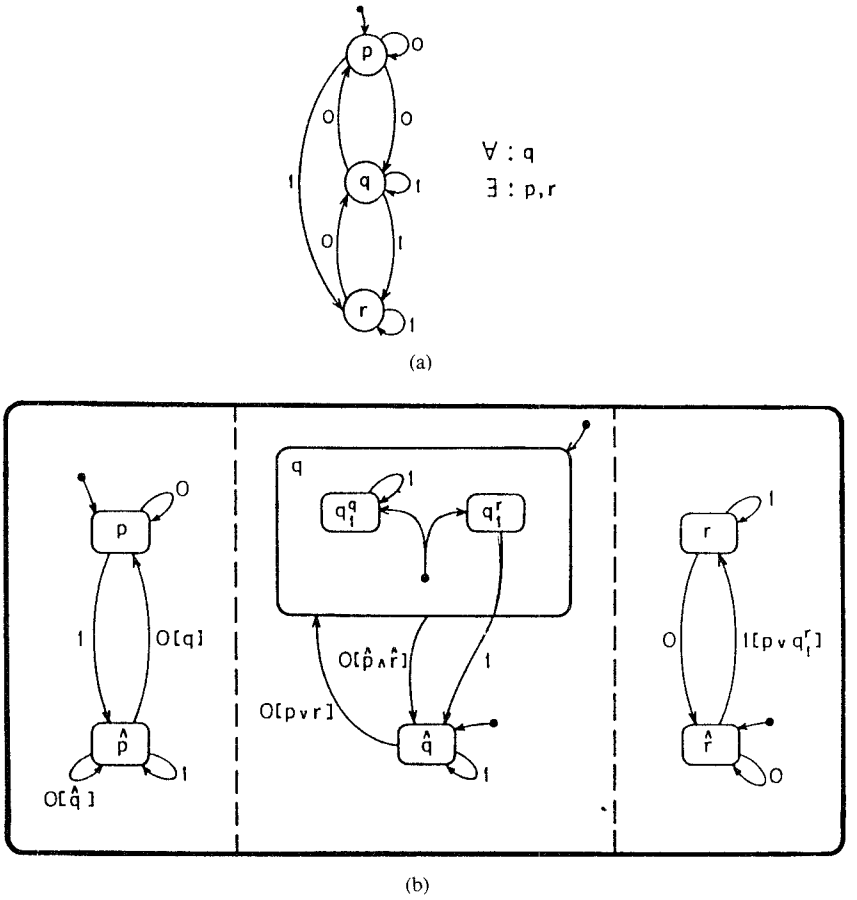

FIG. 4. An AFA and its equivalent ∀-statechart.

p),[8] then N will have the following yes-states (fixing the order a, b, c on Σ): $q^{p,q,p}$, $q^{p,s,p}$, $q^{p,r,p}$, $q^{t,q,p}$, $q^{t,s,p}$, and $q^{t,t,p}$. The intention is to make a "grand" AND-decision up front, when entering the universal state, as to which state will be entered in the next step when any symbol is read. For example, the meaning of $q^{r,s,p}$ is that we have decided to go to r if a arrives, and to s if b arrives. For c, of course, there is no alternative. (In Figure 4 we have omitted q's superscript for the symbol a, since there is but one possibility—like the case for c in the example above—and have added b as a subscript for clarification.)

Now for the transitions. We first define the ones that specify entrance to a state. Here, as before, when a is read as an input, and p's component is in its no-state, it should move into its yes-state if any of p's a-predecessors in the original NFA is in *its* yes-state right now. Formally, for any state $p \in Q$ and symbol $\sigma \in \Sigma$, the simulating machine N will have transitions $(\hat{p}, \sigma, q, p^*)$ and (p, σ, q, p^*), whenever the original machine has a transition (q, σ, p), and q is existential. If q is universal, N's transitions will be $(\hat{p}, \sigma, \gamma, p^*)$ and (p, σ, γ, p^*) instead. Here, p^* stands for each of p's yes-states in N (which

[8] Note that since M has only one component, we can assume, without loss of generality, that there are no condition formulas on transitions.

will be p itself, with no superscripts, if p is existential), and $\gamma \in \Gamma$ is the disjunction of all yes-states of q whose superscript has p in the position corresponding to σ.

As to when states are left, if p is in its yes-state, it should move to its no-state when a arrives if *all* of its a-predecessors are in their no-states. In our case, however, the predecessors may be in the "wrong" kind of yes-states, so that we have to formulate the guarding condition accordingly. Formally, for any state $p \in Q$ and symbol $\sigma \in \Sigma$, the simulating machine N will have the transitions $(p^*, \sigma, \gamma, \hat{p})$ and $(\hat{p}, \sigma, \gamma, \hat{p})$, where p^* is as before, and γ is a conjunction consisting of (i) $\hat{q}$, for any existential q for which the original NFA M has a transition (q, σ, p), and (ii) $\neg q^\tau$, for each universal q for which M has a transition (q, σ, p), and for any superscript τ that has p in the position corresponding to σ.

It is not too difficult to see that the total size of N is bounded by a polynomial in n. Actually, it is bounded by $O(n^{m+1})$, so that the degree of the polynomial depends on the size of the alphabet.[9]

The simulation of (E, A) by (E, C) is dual. □

As a corollary, by tracing one vertical line followed by an upward dashed diagonal, we have exponential upper bounds for the four horizontal lines in the upper portion of Figure 2, in analogy with what was known for the lower portion thereof:

COROLLARY 3. *Let* ξ *be* $\varnothing$ *or* E. *Then* $(\xi, \mathrm{A}, \mathrm{C}) \dashrightarrow (\xi, \mathrm{C})$. *The same holds with* A *and* E *exchanged.*

Also, the upper exponential bounds going downward along the four dashed diagonals and along the doubly dashed diagonal are obtained easily by moving down a vertical line and noticing that the resulting machines are special cases of the target ones:

COROLLARY 4

(1) *Let* ξ *be* $\varnothing$ *or* A. *Then* $(\xi, \mathrm{C}) \dashrightarrow (\xi, \mathrm{E})$. *The same holds with* A *and* E *exchanged.*
(2) $\mathrm{C} \dashrightarrow (\mathrm{E}, \mathrm{A})$.

Finally, the exponential upper bound along the upward direction of the doubly dashed diagonal follows by tracing one solid and one dashed line in the figure:

COROLLARY 5. $(\mathrm{E}, \mathrm{A}) \dashrightarrow \mathrm{C}$.

4. *Lower Bounds for the* Σ^* *Case*

In order to establish the exponential lower bounds represented in all the solid lines of Figure 2, as well as the double-exponential ones implicit in the appropriate compound transitive paths, it suffices to establish the *triple*-exponential lower bound for the simulation of (E, A, C)-machines by deterministic finite automata (i.e., $\varnothing$-AUT). All the aforementioned bounds then follow

[9] It is possible to avoid this—at the expense of making the construction a little more complicated—by using separate copies of and-states, one for each letter of the alphabet, before proceeding with a version of the above construction. The details are omitted.

immediately, since any violation would contradict either this triple-exponential lower bound or the previously established upper bounds.

PROPOSITION 6. $(\mathrm{E}, \mathrm{A}, \mathrm{C}) \dddot{\rightarrow} \varnothing$.

PROOF. We exhibit a family of regular sets, K_n, for $n > 0$, such that each K_n is accepted by an (E, A, C)-AUT of size $O(\log^2 n)$ but the smallest DFA accepting it is at least of size 2^{2^n}.

Our sets are very similar to those appearing in Chandra and Stockmeyer, [1976] (which themselves are based upon sets appearing in Meyer and Fischer, [1971]).

$$K'_n = \{(0, 1, \#)^*\#w\#(0, 1, \#)^*\$w \mid w \in \{0, 1\}^n\}.$$

K'_n represents a simple search problem in which a sequence of words over $\{0, 1\}$ is searched for the occurrence of a particular word w of size n that is given as a suffix following a delimiting \$ symbol. In order to make the illustration in Figure 5 somewhat cleaner, we shall work with a variant of this language, in which &0 and &1 replace 0 and 1, respectively. Also, the words contain extra #'s, and end with a $\dashv$ symbol. Let

$$K_n = \{(\&0, \&1, \#\#)^*\#\#w\#\#(\&0, \&1, \#\#)^*\$w \dashv \mid w \in \{\&0, \&1\}^n\}$$

A standard argument as in, e.g., Chandra and Stockmeyer [1976] and Meyer and Fischer [1971] shows that the smallest DFA accepting K_n has at least 2^{2^n} states. This is because, when the \$ is reached, the automaton has to be able to have remembered any possible set of words of length $2n$ over {&0, &1}, and there are 2^{2^n} such possibilities.

On the other hand, there is an (E, A, C)-AUT of size $O(\log^2 n)$ that accepts K_n. Figure 5 illustrates this, by way of a schematic description of an alternating statechart. Nondeterminism is utilized to guess which of the words in the initial sequence is the sought-for w. Then, in the left-hand side branch, universal branching is used to check, in parallel, that the n pairs of locations in the two alleged occurrences of w indeed contain identical 0/1 bits. Each such check is carried out using bounded concurrency as follows. The machine remembers the bit it is looking at (in the first occurrence of w), and in an orthogonal portion it counts up to n, suspending the count after consuming the rest of w (i.e., when a # is reached) and resuming it when the second alleged w is met (i.e., when the \$ is reached). This will cause the machine to reach precisely the corresponding bit in the second occurrence of w, at which point the counting portion enters a special state OK. The checking component now enters a final state iff the bits are identical. The right-hand side branch checks separately, in parallel, that both w's are of length n. Binary counters are used for these tasks, utilizing the ability of the C feature to simulate counting to n in base 2 using $\log n$ yes/no components; carries are simulated by the state-sensing mechanism of the conditions along transitions. See Figure 6. The machine has $O(\log n)$ states, and the conditions in the counters can be each of length $O(\log n)$, yielding a total size of $O(\log^2 n)$. □

We now establish exponential lower bounds on both directions of the doubly dashed diagonal. The bound for the upward direction follows directly from the double-exponential lower bound $(\mathrm{E}, \mathrm{A}) \ddot{\rightarrow} \varnothing$ and the one-exponential upper

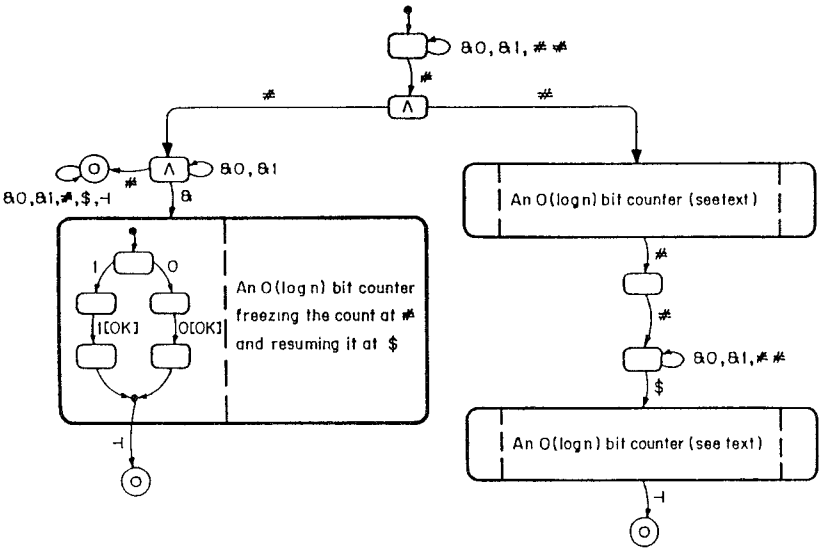

FIG. 5. A logarithmic-size alternating statechart that accepts K_n.

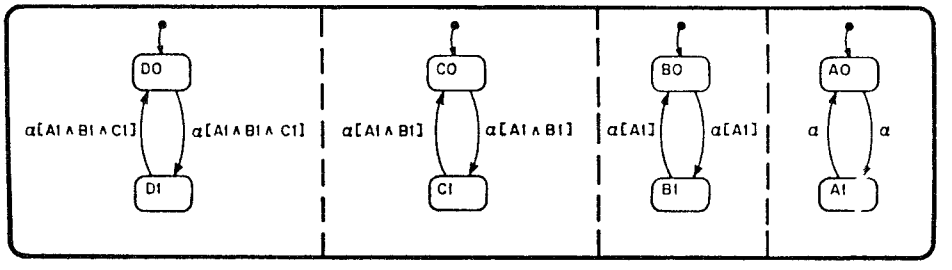

FIG. 6. A four-bit statechart counting the occurrences of α.

bound $C \dot{\to} \varnothing$:

COROLLARY 7. $(E, A) \dot{\to} C$.

The lower bound for the *downward* direction of the doubly dashed diagonal is more subtle. We have to exhibit a set accepted by a polynomial-size deterministic C-AUT, but which requires exponentially many states in an alternating finite automaton.[10] We use a technical lemma relating AFAs to DFAs, which was first stated in Chandra et al. [1981, p. 131].

LEMMA 8. *Let M be an AFA of size n, accepting the language L. There is a DFA with no more than 2^n states that accepts the language L^r, that is, the set consisting of the words of L in reverse.*

Although a proof of the lemma can be gleaned from Chandra et al. [1981], we can prove it directly by considering the reversed sequence of the sets of

[10] This bound was conjectured in the preliminary version of the paper [Drusinsky and Harel, 1988], and was subsequently proved by Hirst [1989].

states that M goes through in an accepting trace. We omit the proof here, as Lemma 21 in Section 6 contains the proof of a (stronger) infinitary version. Now to the lower bound itself:

PROPOSITION 9 [HIRST, 1989]. $C \underset{\cdot}{\rightarrow} (E, A)$.

PROOF. We use essentially the reversed version of the sets used in the proof of Proposition 6. Let

$$X_n = \{w\$(0, 1, \#)^*\#w\#(0, 1, \#)^* \mid w \in \{0, 1\}^n\}.$$

Since the reversed language, X_n^r, is exactly the set K_n' of Proposition 6, it follows that the smallest DFA accepting it is of size at least 2^{2^n}. Thus, by Lemma 8, the smallest AFA accepting the nonreversed version, X_n must have at least 2^n states. To complete the proof, we describe the operation of a C-AUT (e.g., a deterministic statechart) that accepts X_n. It "stores" the first word it sees, w, in binary form (by n orthogonal yes/no components), and then checks each subsequent word, symbol by symbol, for equality with w. It is easy to construct this machine to be of linear size. □

Turning now to the four singly dashed diagonals of Figure 2, we notice that the exponential lower bounds in the downward direction follow immediately from Proposition 9, since E-machines and A-machines are special cases (E, A)-machines, and C-machines are special cases of both (E, C)-machines and (A, C)-machines. Thus, in particular, while nondeterminism can be replaced by bounded concurrency without essential blowup in size (Proposition 2), the converse is not true; the C feature is strictly stronger than E or A. This completes the lower bounds discussed in the Introduction.

An additional set of bounds that are of interest are the "sideway" diagonals that involve going from (A, C) to E and from (E, C) to A. The situation we now establish for them should be contrasted with the third possible comparison of two of the features with the third one, namely the comparison of (E, A) and C, where we had exponential lower bounds in both directions. Here we have double-exponential upper and lower bounds going downward and polynomial bounds going upward, which constitutes further evidence of the greater power of the C feature. The upward bounds follow from the polynomial simulations going from E to C and from A to C (Proposition 2). The upper bound of two exponentials going downward is obtained simply by following one horizontal and one dashed diagonal. It remains to prove the double-exponential lower bound:

PROPOSITION 10. $(A, C) \underset{\cdot\cdot}{\rightarrow} E$ *and* $(E, C) \underset{\cdot\cdot}{\rightarrow} A$.

PROOF. We exhibit a family of regular sets, S_n, for $n > 0$, such that each S_n is accepted by an (A, C)-AUT of size $O(\log n)$, but the smallest E-AUT accepting has at least 2^n states. Define

$$S_n = \{w\$w \mid w \in \{0, 1\}^n\}.$$

The following simple argument shows that any NFA that accepts S_n must have at least 2^n states. Given such an NFA, associate with each w in $\{0, 1\}^n$ a state, by choosing some accepting computation of the word $w\$w$ and singling out the state reached after reading the first w. Call it $q(w)$. Since there are 2^n

different w's, if the automaton had less than 2^n states there would be $w \neq u$ with $q(w) = q(u)$. It is easy to see that the word $w\$u$ would then be accepted.

In contrast, an (A, C)-AUT can be constructed to accept S_n, similar to the (E, A, C)-AUT in the proof of Proposition 6. It first uses AND-states to choose a symbol in the initial w to check. It then counts up to $n + 1$ with $\log n$ orthogonal components, causing the machine to reach the corresponding symbol in the second word; the checking itself takes a constant number of states. In parallel, similar counts are used to make sure that the two words are of length n.

The case with E and A is proved similarly, using the complement of S_n. □

Finally, we would like to show that we have lower bounds in the *reverse* directions of all the solid and singly dashed arrows in Figure 2. These are the arrows for which we have polynomial (mostly linear) upper bounds. For example, we would like to show that E does not always decrease the size exponentially relative to ∅. It is possible to establish these bounds for a uniformly fixed alphabet, such as {0, 1}, as in our previous lower bounds (see Armoni [1991]). We leave this more satisfactory version to the reader; here we prove the bounds trivially using alphabets that grow with n. The following proposition covers all the cases:

PROPOSITION 11. *There is a family of regular sets F_n, for $n > 0$, such that each F_n is accepted by a DFA of size $O(n)$, but the smallest* (E, A, C)-AUT *accepting it is of size at least n.*

PROOF. We use simple one-word languages. Let

$$F_n = \{a_1 a_2 \cdots a_n\}.$$

Even our most powerful machine, an (E, A, C)-AUT, requires each of the a_i to appear on at least one edge, otherwise it can easily be shown to misbehave. A trivial DFA with n states accepts F_n. □

Discussion. Since the C feature adds an exponential amount of succinctness to a deterministic finite automaton, one would expect the standard decision problems to behave differently on C-AUT. This is indeed true. It is possible to show, for example, that the emptiness problem for C-AUT and the determinism problem for (E, C)-AUT are both PSPACE-complete, and that the equivalence problem for C-AUT is NP-complete. In contrast, of course, for DFAs these are in PTIME.

5. *Upper Bounds for the Σ^ω Case*

In this section and the next, we assume that our machines work on inputs from Σ^ω. The results are essentially as in the finite word case, but involve the two acceptance criteria, as illustrated in Figure 3. (In the figure, lines with no R or S mark apply to both Rabin and Streett criteria.) We make heavy use of the results of Safra [1988; 1992] providing determinization and combined determinization-and-complementation results for automata on infinite words with single-exponential blowup.

First, we establish the exponential upper bounds along the solid vertical lines of Figure 3.

PROPOSITION 12. *Let* ξ *be any subset of* {E, A}. *Then* $(\xi, \mathrm{C}) \dot{\to}_{\mathrm{R,S}} \xi$.

PROOF. The reasoning in the proof of Proposition 1 holds here too. Note that the accepting pairs in Ω have to be written in terms of the new states, which are tuples of states taken from the orthogonal components of the original machine. □

We now use Safra's results [1988, 1992] to establish the exponential upper bounds along two of the bottom horizontal lines of Figure 3:

PROPOSITION 13. $\mathrm{E} \dot{\to}_{\mathrm{R,S}} \varnothing$ *and* $\mathrm{A} \dot{\to}_{\mathrm{R,S}} \varnothing$.

PROOF. That $\mathrm{E} \dot{\to}_{\mathrm{R}} \varnothing$ is exactly the main result in Safra [1988] to the effect that nondeterminism can be eliminated in Rabin automata with only a single exponential growth in size. (The result in Safra [1988] transforms from Büchi automata to Rabin automata, but one first carries out the easy polynomial transformation of nondeterministic Rabin machines to nondeterministic Büchi machines; see, for example, Lemma 5 of Safra [1988].) That $\mathrm{A} \dot{\to}_{\mathrm{S}} \varnothing$ follows from the following easily proved duality:

> Let $L \subseteq \Sigma^{\omega}$ be R-accepted by the nondeterministic automaton M with acceptance set Ω. If N is the automaton M, but viewed as an A-ωAUT (that is, its states are all considered AND-states), and with the same acceptance set Ω, then N S-accepts $\overline{L}$. The same holds with R and S exchanged.

For $\mathrm{E} \dot{\to}_{\mathrm{S}} \varnothing$ (and by the same duality, also $\mathrm{A} \dot{\to}_{\mathrm{R}} \varnothing$), we use Safra's determinization-and-complementation result for Streett automata [Safra, 1992]. It establishes, with a single exponential blow-up, a transformation from a nondeterministic Streett automaton to a deterministic Rabin automaton that accepts the complement of the original language. The dual automaton (in the sense of the above duality) S-accepts the original language. □

Here are the other two bottom horizontal lines.

PROPOSITION 14. $(\mathrm{E}, \mathrm{A}) \dot{\to}_{\mathrm{R.S}} \mathrm{E}$ *and* $(\mathrm{E}, \mathrm{A}) \dot{\to}_{\mathrm{R.S}} \mathrm{A}$.

PROOF. First, $(\mathrm{E}, \mathrm{A}) \dot{\to}_{\mathrm{R}} \mathrm{E}$.

Let M be a given (E, A)-ωAUT operating over the alphabet Σ, and with state set Q. We first transform M into an A-ωAUT M' that operates over a richer alphabet. Define $\Sigma' = \Sigma \cup (\Sigma \times Q \times Q)$. M' has the same state set Q, the same start state and the same acceptance pairs. Its transitions are obtained from those of M in the following way. Transitions emanating from universal states are included in M' without change. For an existential state p, if M contains a transition leading to state q for input symbol a, then in M' the symbol triggering the transition from p to q will be the triple (a, p, q). Thus, we have replaced the nondeterminism of M by deterministic choices, in which the source and target states have been "remembered" alongside the input symbol that caused the transition. Note also that the size of M' is polynomial in the size of M.

The *projection* of a word $x \in \Sigma'^{*}$ over Σ, denoted by x_{Σ}, is obtained from x simply by dropping the state components from any triple appearing in x. Let $L \subseteq \Sigma^{*}$ be the language R-accepted by M, and let $L' \subseteq \Sigma'^{*}$ be the language R-accepted by M'. It is straightforward to show that for any $y \in \Sigma^{*}$, $y \in L$ iff there is some $x \in L'$, with $y = x_{\Sigma}$. The details of this are left to the reader.

We now dualize our Rabin A-ωAUT, M', considering its states to be existential and its acceptance mechanism to be Streett. The new automaton S-accepts $\overline{L'}$. Now, we apply Safra's [1992] co-determinization result to this nondeterministic Streett automaton, yielding an $\emptyset$-ωAUT of exponential size that R-accepts L'. Call if N'. The final step is to "project out" the state pairs in the enriched alphabet of N', by simply replacing any triple-symbol (a, p, q) in a transition by a. This yields an E-ωAUT N that R-accepts the original language L.

We can now use this construction (with the help of Safra's results, of course) to obtain the (E, A) $\dot{\rightarrow}_S$ E transformation too. Consider the automaton N' above. It has an exponential number of states, but only a linear number of pairs in its termination condition. In the full version of Safra [1988], Safra shows how to transform a deterministic Rabin automaton into a deterministic Streett automaton using an exponential blow-up in the number of pairs only. In this way, we can obtain an $\emptyset$-ωAUT of exponential size that S-accepts L'. One then projects out the state pairs as above.

The (E, A) $\dot{\rightarrow}_{R,S}$ A transformations follow by duality. □

In analogy with Proposition 2, we now prove polynomial upper bounds for the upward direction of the four dashed diagonal lines of Figure 3. This entails a certain strengthening of Safra's constructions:

PROPOSITION 15

(1) E $\overset{p}{\rightarrow}_R$ C *and* A $\overset{p}{\rightarrow}_S$ C;

(2) (E, A) $\overset{p}{\rightarrow}_R$ (E, C) *and* (E, A) $\overset{p}{\rightarrow}_S$ (A, C).

PROOF. Consider the claim E $\overset{p}{\rightarrow}_R$ C. Given an E-ωAUT of size n, we first carry out the easy transformation into an R-equivalent nondeterministic Büchi automaton, with at most a polynomial increase in size. The idea now is to implement Safra's construction [Safra, 1988] of an exponential-sized R-equivalent $\emptyset$-ωAUT using a polynomial-sized "data structure" that can be described and manipulated by a deterministic C-ωAUT of roughly the same polynomial size. The full details involve tedious programming of a complicated C-ωAUT (e.g., a deterministic statechart), and are omitted in favor of a high-level description.

Let us go through the steps of Safra's construction. Denote by Q the set of states in the given Büchi automaton M. In the new deterministic machine N, the states are ordered trees, each node of which is colored white or green and is labeled with some subset of Q. These trees satisfy the following conditions. The label of a node is a subset of the label of its parent, and the label of the parent, in turn, contains at least one state of Q not present in the labels of any of its child nodes. In this way, the union of the sets labeling the child nodes of any node is a strict subset of the set labeling the parent. Moreover, the sets labeling sibling nodes are pairwise disjoint. Consequently, the tree cannot contain more than $|Q|$ nodes. The start state of N is the tree containing one white node only, labeled by $\{q_0\}$, the start state of M.

Here is how N behaves when in the state given by a tree T if the symbol it sees is a. It first colors all nodes of T white. Next, to any node whose label set intersects F—the set of accepting states of M—we attach a new child node labeled with the said intersection. The new node is to be the rightmost (i.e.,

youngest) child. Now, the transition table δ of the original machine M is applied to each node in T, pointwise to all states in the label, resulting in new label sets of states for each node. A series of corrections to this temporary tree is now carried out. First, if, as a result of this application of δ, we find a state $s \in Q$ appearing in the labels of more than one child of some node, we delete s from all those child nodes except the oldest (i.e., leftmost) one in which it appears. As a result, each state of Q that appears in the tree, appears along a unique path of nodes stretching from the root to some node. Any node whose label becomes empty as a result of these deletions is eliminated from the tree, together with its entire subtree. Finally, if, as a result of all of this, the set labeling some node becomes equal to the union of the sets labeling its child nodes, the child nodes are all deleted from the tree (together with their subtrees), and the node itself is colored green.

As shown in Safra [1988], no more than $2n$ new nodes are ever created during a run of N, where n is the size of M, so that we can attach a unique name to each node from among $1, 2, \ldots, 2n$. The acceptance set of N is now taken to contain all pairs of the form (L_v, U_v), for $v \in \{1, 2, \ldots, n\}$, where L_v is the set of trees that contain the node v colored green, and U_v is the set of trees that do not contain v at all. That this construction works lies at the heart of Safra's proof. In the construction, the new deterministic automaton has the trees as states, so that its size is the number of possible trees (times the size of the names of the nodes), which is exponential in n.

Our goal is to show how this very construction can be implemented by a C-ωAUT of size polynomial in n. The basic idea is to construct the machine to maintain a complex set of orthogonal components that have the capacity to denote all possible trees and the labels and names of their nodes. Figure 7 contains a schematic statechart illustration of the main components in this machine. There is a vertical row of components for each potential node of the tree, and, by the remark made earlier, we need at most $2n$ of them. During each step of the simulating machine, the nodes actually present in the current tree are all left-justified in the figure. Thus, since a node's place in the statechart is not fixed, we hold the name of the node explicitly. In addition, we have the color of each node, and the name of its parent. Finally, the set of states in Q labeling the node is kept too, using orthogonal yes/no components. In addition to the information for each vertical node column, we maintain a table indicating the color of each node by name. This is needed for stating the acceptance criterion.

The initial states (not indicated in the figure) will cause the statechart to initialize to the tree containing a single node (held in the leftmost column) whose only state from Q is q_0. We now have to convince ourselves that we can encode the effect of simulating steps of the deterministic automaton N within this C-ωAUT, with only a polynomially-large transition table. In general, the simulation is carried out by a transition triggered by the input symbol, and which, by entering special states and having other components sense the ones that have been entered, triggers a chain reaction of changes. (Such chain reactions can be set up in the statechart in much the same way as the carry ripples through a counter, as illustrated in Figure 6, without the need for ϵ-transitions.) The various steps in the simulation entail adding several state components to the main parts appearing in Figure 7. These are used mainly to control the order in which the changes are made.

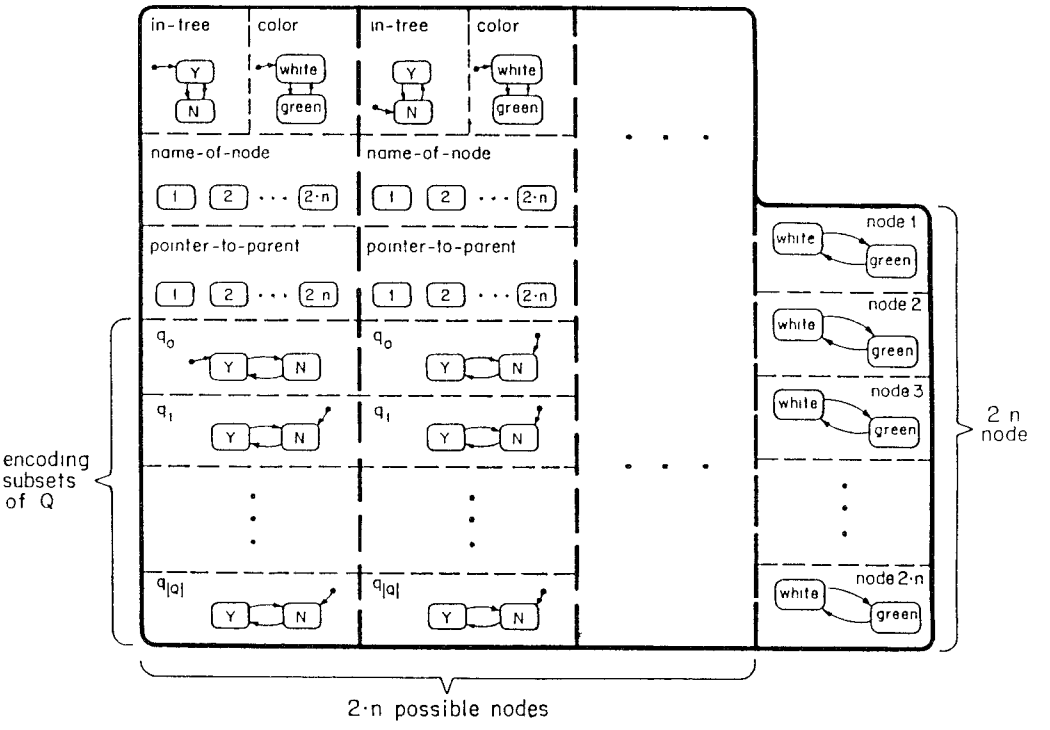

FIG. 7. A polynomial-sized deterministic statechart implementing Safra's [1988] construction.

There are basically three abilities that we must be able to program in order to simulate the entire process: (i) adding a youngest child to a node; (ii) deleting a set of nodes; (iii) copying information from node to node. Some such actions are carried out in each of the vertical columns in order, by rippling triggers from left to right. The most intricate of these is the deletion of nodes, since we have to be able to left-justify the remaining columns. This is done by first calculating, in a new component for each vertical node description i, the unique node k_i that represents the ith node from the left that is to be present in the tree after the deletions. The new component simply uses two counters of states; the first counts up to i, with the second continuously searching for the next vertical component that is in its "present-in-the-tree" state. When this computation is over, the machine triggers the copying routine from left to right, moving the information in vertical column k_i to column i. Applying δ to a node in the tree is done as in the first part of the proof of Proposition 2.

As mentioned, the details of this simulation contain nothing conceptually complicated, and mostly involve just tedious programming. This completes the proof of $E \xrightarrow{p}_R C$. Clearly, $A \xrightarrow{p}_S C$ follows by duality.

Part (2) of the Proposition refers to the upward direction of the "back" dashed arrows in Figure 3. The proof of $E \xrightarrow{p}_R C$ from part (1) involved implementing Safra's construction in [Safra, 1988] efficiently using the C feature. Similarly, proving the claim $(E, A) \xrightarrow{p}_R (E, C)$ involves using the C feature to efficiently implement the proof of Proposition 14, the heart of which

is Safra's construction in [Safra, 1992] (the (E, A) $\overset{p}{\rightarrow}_S$ (A, C) claim will then also follow by duality).

As in the proof of Proposition 14, we first enrich the alphabet from Σ to Σ', construct the polynomially-sized Rabin A-ωAUT, and promptly view it as a Streett E-ωAUT accepting some language L over Σ'. We now have to mimic Safra's codeterminization procedure from Safra [1992] to obtain a polynomially-sized C-ωAUT (e.g., a deterministic statechart) that R-accepts $\overline{L}$. The proof here is in essence just as before. One carefully follows Safra's construction, using a "data structure" of polynomial size. This has to be done for the constructions in both Theorem 1 and Lemma 3 of Safra [1992]. In fact, in contrast to Safra [1988], the formal constructions in Safra [1992] are preceded by informal descriptions in which the simulating machine is viewed as "a program with bounded memory and some infinitary acceptance condition" [Safra, 1992, Sect. 3]. Together with the points made in the proof of part (1) above, regarding ways to control the order in which changes are made in the "data structure" and to ripple triggers from one part thereof to another, the descriptions in Safra [1992] are sufficient to establish our claim. Here too we leave the programming details to the reader. □

As a corollary, by tracing a vertical line and an upward dashed diagonal, we obtain exponential upper bounds for the four top horizontal lines of Figure 3:

COROLLARY 16

(1) (E, C) $\dot{\rightarrow}_R$ C *and* (A, C) $\dot{\rightarrow}_S$ C;
(2) (E, A, C) $\dot{\rightarrow}_R$ (E, C) *and* (E, A, C) $\dot{\rightarrow}_S$ (A, C).

Also, the exponential bounds going downward along the four dashed diagonals and along the doubly dashed diagonal in Figure 3 are obtained immediately by moving down a vertical line and noticing that the resulting machines are special cases of the target ones:

COROLLARY 17

(1) C $\dot{\rightarrow}_R$E, C $\dot{\rightarrow}_S$A, (E, C) $\dot{\rightarrow}_R$ (E, A), *and* (A, C) $\dot{\rightarrow}_S$ (E, A);
(2) C $\dot{\rightarrow}_{R,S}$ (E, A).

The R and S upper bounds along the upward direction of the doubly dashed diagonal follow by tracing a solid and a dashed line in Figure 3:

COROLLARY 18. (E, A) $\dot{\rightarrow}_{R,S}$ C.

6. *Lower Bounds for the Σ^ω Case*

The lower bounds here are also as in the finite words case. As in Section 4, we shall first prove a triple-exponential lower bound for the transition from (E, A, C)-ωAUT to $\emptyset$-ωAUT. This will establish the exponential lower bounds along the solid lines in Figure 3, as well as the double-exponential ones implicit in the appropriate compound transitive paths.

PROPOSITION 19. (E, A, C) $\dot{\dot{\rightarrow}}_{R,S}\emptyset$.

PROOF. Recalling the sets K_n of Proposition 6, we take J_n to be simply $K_n \cdot 0^\omega$. That a $\emptyset$-ωAUT requires 2^{2^n} states to accept J_n follows the same argument used for the finite word case, since, when the \$ symbol is reached, the machine has to be able to have remembered any possible set of words of length n over {0, 1}. Also, to R-accept or S-accept J_n by a logarithmic-sized

(E, A, C)-ωAUT, simply use the machine of Proposition 6 to first confirm that the portion up to the $\dashv$ symbol is in K_n, and then an additional state will be used to read the 0^ω and accept according to the criterion of choice. □

As in the finite word case, the exponential bound for the upward direction of the doubly dashed diagonal follows directly from the double-exponential (E, A) $\dddot{\rightarrow}_{R,S} \varnothing$ lower bounds and the one-exponential C $\dot{\rightarrow}_{R,S} \varnothing$ upper bounds:

COROLLARY 20. (E, A) $\dddot{\rightarrow}_{R,S}$ C.

The proof of the lower bound for the downward direction of the doubly dashed diagonal also follows along the lines of the proof in Section 4. In particular, we first prove an appropriately amended version of Lemma 8:

LEMMA 21. *Let $L \subseteq \Sigma^*$ be a regular set, and let M be an* (E, A)-ωAUT *of size n that* R*-accepts the ω-regular set $L \cdot 0^\omega$. There is a $\varnothing$-ωAUT N of size at most $2^{O(n)}$ that* R*-accepts the set $L' \cdot 0^\omega$. The same holds for* S*-acceptance.*

PROOF. The states of N are taken to be subsets of those of M. Let V denote the set of states of M with the property that if M is started in such a state on the word 0^ω, it accepts. The start state of N will be the set V. Define the transitions of N so that upon seeing the symbol $a \in \Sigma$ when in state U it enters the unique state W defined as follows: For a state w of M, we put w in W iff at least one a-successor of w (when considered as a state of M) is in U if w is existential, and all a-successors of w are in U if w is universal. In addition, we set things up in N so that when it reads 0^ω starting in any state U that contains the start state of M, it accepts. This can be done very easily using either the Rabin or the Streett criteria.

It is now straightforward to show that N accepts a word $x \cdot 0^\omega$ iff $x^r \in L$, which is exactly when M accepts $x^r \cdot 0^\omega$. □

Now for the lower bound:

PROPOSITION 22. $C \dddot{\rightarrow}_{R,S} (E, A)$.

PROOF. Again, we use a simple infinitary version of the sets X_n of Proposition 9. Let Y_n be defined as $X_n \cdot 0^\omega$. That no (E, A)-ωAUT with less than 2^n states R-accepts or S-accepts Y_n follows from Lemma 21 and the argument in the proof of Proposition 19. To accept it with a linear-sized C-ωAUT, a new state is added to the statechart described in the proof of Proposition 9, to accept if the rest of the word is 0^ω. □

The lower bounds along the dashed diagonals now follow as in the finite word case.

ACKNOWLEDGMENTS. We would like to thank Rafi Heiman, Tirza Hirst, Charanjit Jutla, Nils Klarlund, Oded Maler, Danny Raz, Roni Rosner, Shmuel Safra, and Moshe Vardi for many stimulating discussions related to the material presented here. We are particularly grateful to Charanjit Jutla, Danny Raz, and Moshe Vardi for their help in establishing Proposition 14. We decided to adopt the definitions of C-machines in the M.Sc. theses of Roy Armoni and Noa Globerman for the definitions we use here. The two referees provided very helpful comments.

REFERENCES

ARMONI, R. 1991. On the succinctness gained by bounded cooperative concurrency. M.Sc dissertation. The Weizmann Institute of Science, Rehovot, Israel.

BOOK, R. V., AND GREIBACH, S. A. 1970. Quasi-realtime languages. *Math. Syst. Theory 4*, 97–111.

CHANDRA, A. K., KOZEN, D., AND STOCKMEYER, L. J. 1981. Alternation. *J. ACM 28*, 1 (Jan.), 114–133.

CHANDRA, A. K., AND STOCKMEYER, L. J. 1976. Alternation. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science.* IEEE Press, New York, pp. 98–108.

CHOUEKA, Y. 1974. Theories of automata on ω-tapes: A simplified approach. *J. Comput. Syst. Sci. 8*, 117–141.

DRUSINSKY, D., AND HAREL, D. 1988. On the power of cooperative concurrency. In *Proceedings of Concurrency '88.* Lecture Notes in Computer Science, vol. 335. Springer-Verlag, New York, pp. 74–103.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York.

GLOBERMAN, N., AND HAREL, D. 1994. Succinctness results for multi-pebble automata. In *Proceedings of the 21st International Colloquium on Autmata, Languages, and Programming.* Lecture Notes in Computer Science, Springer Verlag, New York (to appear).

HAREL, D. 1987. Statecharts: A visual formalism for complex systems. *Sci. Comput. Prog. 8*, 231–274.

HAREL, D. 1989. A thesis for bounded concurrency. In *Proceedings of the 14th Symposium on Mathematical Foundations of Computer Science.* Lecture Notes in Computer Science, vol. 379. Springer-Verlag, Berlin, Germany, pp. 35–48.

HAREL, D., ROSNER, R., AND VARDI, M. 1990. On the power of bounded concurrency III: Reasoning about programs. *Proceedings of the 5th Symposium on Logic in Computer Science.* IEEE Press, New York, pp. 479–488.

HIRST, T. 1989. Succinctness results for statecharts. M.Sc. dissertation. Bar-Ilan Univ., Ramat Gan, Israel (in Hebrew).

HIRST, T., AND HAREL, D. 1994. On the power of bounded concurrency II: Pushdown automata. *J. ACM 41*, 3 (May), 540–554.

HOARE, C. A. R. 1978. Communicating sequential processes. *Commun. ACM 21*, 8 (Nov.), 666–677.

HROMKOVIĈ, J., INOUE, K., ROVAN, B., SLOBODOVÁ, A., TAKANAMI, I., AND WAGNER, K. W. 1992. On the power of one-way synchronized alternating machines with small space. *Int. J. Found. Comput. Sci. 3*, 65–79.

KOZEN, D. 1976. On parallelism in Turing machines. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science.* IEEE Press, New York, pp. 81–88.

KOZEN, D. 1977. Lower bounds for natural proof systems. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science.* IEEE Press, New York, pp. 254–266.

MEYER, A. R., AND FISCHER, M. J. 1971. Economy of description by automata, grammars, and formal systems. In *Proceedings of the 12th IEEE Symposium on Switching and Automata Theory.* pp. 188–191.

MILNER, R. 1980. A calculus of communicating systems. In *Lecture Notes in Computer Science*, vol. 94. Springer-Verlag, New York.

RABIN, M. O. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. AMS 141*, 1–35.

RABIN, M. O., AND SCOTT, D. 1959. Finite automata and their decision problems. *IBM J. Res. 3*, 115–125.

REISIG, W. 1985. *Petri Nets: An Introduction.* Springer-Verlag, Berlin.

SAFRA, S. 1988. On the complexity of ω-automata. In *Proceedings of the 29th Symposium on Foundations of Computer Science.* IEEE Press, New York, pp. 319–327.

SAFRA, S. 1992. Exponential determinization for ω-automata with strong fairness acceptance condition. In *Proceedings of the 24th ACM Symposium on Theory of Computing* (Victoria, B.C., Canada, May 4–6). ACM, New York, pp. 275–282.

SAVITCH, W. J. 1970. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci. 4*, 177–192.

SLOBODOVÁ, A. 1988. On the power of communication in alternating machines. In *Proceedings of the 13th Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, vol. 324. Springer-Verlag, Berlin, pp. 518–528.

STREETT, R. S. 1982. Propositional dynamic logic with converse. *Inf. Cont. 54*, 121–141.

RECEIVED JULY 1988; REVISED APRIL 1992; ACCEPTED NOVEMBER 1993