SIAM J. CONTROL AND OPTIMIZATION
Vol. 32, No. 4, pp. 1075–1097, July 1994

011

CONTROL OF INFINITE BEHAVIOR OF FINITE AUTOMATA*

J. G. THISTLE† AND W. M. WONHAM‡

Abstract. A problem in the control of automata on infinite strings is defined and analyzed. The key to the investigation is the development of a fixpoint characterization of the "controllability subset" of a deterministic Rabin automaton, the set of states from which the automaton can be controlled to the satisfaction of its own acceptance condition. The fixpoint representation allows straightforward computation of the controllability subset and the construction of a suitable state-feedback control for the automaton. The results have applications to control synthesis, automaton synthesis, and decision procedures for logical satisfiability; in particular, they represent a direct, efficient and natural solution to Church's problem, the construction of winning strategies for two-player zero-sum ω-regular games of perfect information, and the emptiness problem for automata on infinite trees.

Key words. discrete-event systems, synthesis problems, Church's problem, tree automaton emptiness, ω-languages, ω-automata, ω-regular games

AMS subject classifications. 93B50, 03D05, 68Q68, 03B70, 03B25, 03B45, 90D05, 68Q60, 03C30, 03C50, 03B35

1. Introduction.

This paper and the companion article [42] outline a theory of the control of infinite behavior of discrete-event systems. Based on [43], the articles extend some of the fundamental results of the finitary *supervisory control theory* of Ramadge and Wonham (see [35]). The present paper focuses on a key computational problem that arises not only in connection with control but also in several other contexts within the study of infinite behavior of discrete-event systems. As formulated here, the problem concerns the control of finite automata on infinite strings, commonly termed finite *ω-automata*.

A survey of the theory of ω-automata is given in [44]. Like their more familiar counterparts on finite strings, ω-automata consist of transition structures and accompanying acceptance conditions. However, whereas acceptance conditions for finite strings depend on the state *last* entered by the automaton upon reading a symbol string, the criterion for infinite strings instead involves the set of states entered *infinitely often* during the processing of the string. This article is specifically concerned with *Rabin* acceptance conditions (see §3).

Though relatively new to control theory, ω-automata are well-established tools in the theory of discrete-event systems. First introduced by Büchi as a means of deciding satisfiability of certain logical formulas [3] and by Muller as a natural means of describing infinite behavior of asynchronous switching circuits [27], they have since been applied to numerous studies of digital hardware, computer software and the associated logics (see, for example, [44], [12], [22], [23], [48], [46], [29], [1]), and, to a more limited extent, discrete-event control systems (see [2], [17], [36], [18], [40]).

* Received by the editors July 19, 1991; accepted for publication (in revised form) January 20, 1993.

† Département de génie électrique et de génie informatique, Ecole Polytechnique de Montréal, C.P. 6079, succ. Centre-ville, Montréal, Canada H3C 3A7. The work of this author was partially supported by Natural Sciences and Engineering Research Council of Canada grant OGP0007399 and a Postdoctoral Fellowship.

‡ Systems Control Group, Department of Electrical Engineering, University of Toronto, Toronto, Canada M5S 1A4. The work of this author was partially supported by Natural Sciences and Engineering Research Council of Canada grant OGP0007399.

Automata are traditionally interpreted as information-processing devices, "reading" strings of symbols (or inputs of other forms), executing a state transition in response to each symbol, and either accepting or rejecting strings according to their acceptance conditions. From the perspective of control, however, they are more naturally viewed as dynamic systems that spontaneously execute sequences of state transitions, "generating" strings of symbols as they do so, the acceptance condition distinguishing those strings that in some sense represent desirable behavior [34]. This is the view that we adopt here. Moreover, it is assumed here that, at any point in the system's operation, it is possible to restrict the set of symbols that may be generated to any one of a given family of subsets of the complete alphabet; this represents a means of control over the state transitions of the automaton. The article examines the computation of the "controllability subset" of a deterministic Rabin automaton (the set of states from which the automaton can be controlled to the satisfaction of its own acceptance condition) and the construction of corresponding control strategies.

This problem arises naturally in the study of the supervisory control of infinite behavior of discrete-event systems; indeed, the results of the current article are applied to effective supervisor synthesis in [42]. The same problem, however, also occurs in several other branches of the theory of discrete-event systems. In its earliest formulation, it represents a basic paradigm for program synthesis: *Church's problem* is that of constructing an automaton whose infinite input-output behavior satisfies a given logical formula [6], [7]. As many of the relevant logical languages are equivalent in expressiveness to finite ω-automata, Church's problem is often formulated in purely automata-theoretic terms, which make it formally equivalent to the problem studied here [24], [4], [33], [29], [28]. The first comprehensive solution to Church's problem exploited its equivalence to the problem of constructing winning strategies for two-person, zero-sum, and ω-regular games of perfect information [24], [4]. Simpler solutions have since been obtained through reduction to a further equivalent problem, the so-called *emptiness problem for automata on infinite trees* [44], [33], [29], [1], [49], which represents an important means of deciding satisfiability for a variety of logics, including many propositional logics of programs [12], [31], [32], [38], [47].

The present control-theoretic formulation admits a particularly direct and natural solution. First, the controllability subset is defined "operationally" in terms of the infinite strings generated by the automaton under suitable control. It is then shown that this operational definition can be replaced by a "denotational" one, a more direct characterization of the controllability subset as a certain fixpoint of an "inverse dynamics operator," which depends only on the one-step dynamics of the controlled automaton. The computation of this fixpoint is straightforward, and intermediate results of the calculation can be used to compute a suitable control in the form of a state feedback map. This approach is essentially optimal in computational complexity.

Section 2 presents some preliminaries concerning extremal fixpoints of monotone operators. In §3 the controllability subset is formally defined. Section 4 discusses some operations on automata that allow for structural induction in later definitions and proofs. The "inverse dynamics" and "p-reachability" operators of a deterministic Rabin automaton are then defined in §5. In §6 the controllability subset is characterized as a certain fixpoint of these operators. The computational complexity of deciding membership of a state in the controllability subset is examined in §7, and it is shown that straightforward computation of the fixpoint is essentially optimal in this respect. The results of the article are illustrated with an example in §8 and compared with earlier work in §9.

One of the advantages of the methods of the article is its extensibility to control under liveness assumptions. The case of liveness assumptions represented by Büchi acceptance conditions is outlined in [41], where the results of the present article are also summarized. More general forms of liveness assumptions will be treated in future reports.

2. Preliminaries: Fixpoints of monotone operators. The methods of this paper require special notation for extremal fixpoints of monotone operators. This section presents a suitable calculus based on the use of "fixpoint quantifiers." Originally applied to the denotational semantics of recursion (see [8]), such quantifiers have also been used to extend the power of various logics of programs [10], [30], [21], [45].

Our definitions of monotonicity and continuity and our presentation of the fundamental results of Tarski and Knaster (Theorem 2.1) are adapted from those of [16].

2.1. Monotone and continuous operators. A k-ary *operator* on a power set 2^X is a map $f : (2^X)^k \to 2^X$.

An operator is *monotone* if it preserves inclusion, that is,

$$X_i \subseteq X_i' \Longrightarrow f(X_1, \ldots, X_i, \ldots X_k) \subseteq f(X_1, \ldots, X_i', \ldots X_k), \qquad 1 \leq i \leq k.$$

An operator is $\cup$-*continuous* if, for any i, $1 \leq i \leq k$, and any nondecreasing sequence $X_i^0 \subseteq X_i^1 \subseteq X_i^2 \cdots$,

$$\bigcup_{j=0}^{\infty} f(X_1, \ldots, X_i^j, \ldots, X_k) = f\left(X_1, \ldots, \bigcup_{j=0}^{\infty} X_i^j, \ldots, X_k\right).$$

An operator is $\cap$-*continuous* if, for any i, $1 \leq i \leq k$ and any nonincreasing sequence $X_i^0 \supseteq X_i^1 \supseteq X_i^2 \cdots$,

$$\bigcap_{j=0}^{\infty} f(X_1, \ldots, X_i^j, \ldots, X_k) = f\left(X_1, \ldots, \bigcap_{j=0}^{\infty} X_i^j, \ldots, X_k\right).$$

Both $\cup$-continuity and $\cap$-continuity imply monotonicity; for operators on finite power sets, the reverse implications hold.

Monotonicity is important for our purposes because it implies the existence of extremal fixpoints (in the sense of set inclusion). The continuity properties provide a convenient means of computing such fixpoints (see Theorem 2.1 below).

2.2. A fixpoint calculus. The expressions of our fixpoint notation consist of monotone operators applied to subsets and the "fixpoint quantifiers" μ and ν that quantify over subsets. Expressions of the form

$$\mu Y.\ \phi(Y) \qquad \text{(respectively, } \nu Y.\ \phi(Y)\text{)}$$

represent the least (respectively, greatest) $Y \subseteq X$ such that $Y = \phi(Y)$, in other words, the least (respectively, greatest) fixpoints of the operator that maps every $Y \subseteq X$ to $\phi(Y)$. The question of the existence of such fixpoints is dealt with below.

- For any $Y' \subseteq X$,

$$\mu Y.\ (Y' \cup Y) = Y'.$$

The dual result is

$$\nu Y.\ (Y' \cap Y) = Y'.$$

• Adding fixpoint quantifiers to the expressions in the previous example, we find that

$$\nu Y'.\ \mu Y.\ (Y' \cup Y) = \nu Y'.\ Y' = X$$

and, dually,

$$\mu Y'.\ \nu Y.\ (Y' \cap Y) = \mu Y'.\ Y' = \emptyset.$$

2.3. Fixpoint lemmata. The basic results on extremal fixpoints of monotone operators follow from more general theorems of Tarski and Knaster.[1]

THEOREM 2.1 (Tarski–Knaster). *Let $f : 2^X \to 2^X$ be a monotone operator on X. Then f has least and greatest fixpoints; in fact,*

(i) $\mu Y.\ f(Y) = \bigcap\{Y' \subseteq X :\ Y' = f(Y')\} = \bigcap\{Y' \subseteq X :\ Y' \supseteq f(Y')\}$,

(i′) $\nu Y.\ f(Y) = \bigcup\{Y' \subseteq X :\ Y' = f(Y')\} = \bigcup\{Y' \subseteq X :\ Y' \subseteq f(Y')\}$,

(ii) *If f is $\cup$-continuous then* $\mu Y.\ f(Y) = \bigcup_{i=0}^{\infty} f^i(\emptyset)$,

(ii′) *If f is $\cap$-continuous then* $\nu Y.\ f(Y) = \bigcap_{i=0}^{\infty} f^i(X)$

(where f^i denotes the i-fold composition of f with itself).

LEMMA 2.2. *Let $f_1,\ f_2 : 2^X \to 2^X$ be monotone operators on X. If*

$$f_1(Y) \subseteq f_2(Y) \quad \forall Y \subseteq X,$$

then (a) $\mu Y.\ f_1(Y) \subseteq \mu Y.\ f_2(Y)$ *and* (b) $\nu Y.\ f_1(Y) \subseteq \nu Y.\ f_2(Y)$.

Theorem 2.1 guarantees that operators have extremal fixpoints, provided that they are monotone. Monotonicity is clearly preserved under composition of operators, and Lemma 2.2 shows that it is preserved under fixpoint quantification.[2] Together, these results imply that the semantics of our calculus are well defined.

3. Controllability subsets of Rabin automata. Before defining Rabin automata and their controllability subsets, we establish some notation for formal languages.

Σ^* denotes the set of all finite strings (including the *empty string*, denoted by 1, having length 0) over some finite symbol alphabet Σ; Σ^ω denotes the set of (countably) infinite strings over Σ; Σ^∞ denotes $\Sigma^* \cup \Sigma^\omega$. A *language* is any set of strings over Σ; in particular, an *ω-language* is a subset of Σ^ω.

A finite string $k \in \Sigma^*$ is a *prefix* of $v \in \Sigma^\infty$ if k is an initial substring of v; in this case, we write $k \leq v$; if k is not identical to v (i.e., if k is a *proper* prefix), we may write $k < v$. For any string $v \in \Sigma^\infty$, we let $\mathrm{pre}(v)$ denote its set of (finite) prefixes, that is, $\mathrm{pre}(v) := \{k \in \Sigma^* : k \leq v\}$.

A *Rabin automaton* [26], [33] is a 5-tuple

$$\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$$

[1] See [39]. For more information on the history of these results, see [25]. The authors are grateful to Karen Rudie for pointing out this second reference.

[2] That is, if $f : (2^X)^k \longrightarrow 2^X$ is monotone, then so is the operator $f_1^\mu : (2^X)^{k-1} \longrightarrow 2^X$ defined by

$$f_1^\mu : (Y_2, \ldots, Y_k) \mapsto \mu Y_1.\ f(Y_1, Y_2, \ldots, Y_k),$$

and similarly for greatest fixpoints.

consisting of: an *alphabet* Σ, a finite *state set* X, a *transition function* $\delta : \Sigma \times X \longrightarrow 2^X$, an *initial state* x_0, and a *family of accepting pairs* $(R_p, I_p) \in 2^X \times 2^X$ with index set P.

It is convenient to extend δ to a map $\delta : \Sigma^* \longrightarrow 2^X$ according to

$$(1, x) \overset{\delta}{\mapsto} x,$$

$$(k\sigma, x) \overset{\delta}{\mapsto} \bigcup\{\delta(\sigma, x_k) : x_k \in \delta(k, x)\} \quad \forall\, k \in \Sigma^*,\ \sigma \in \Sigma.$$

A *path* on $\mathcal{A}$ of a string $v \in \Sigma^\infty$ is a total function $\pi : \mathrm{pre}(v) \longrightarrow X$ such that

$$\pi(1) = x_0 \quad \forall\, k \in \mathrm{pre}(v),\ \sigma \in \Sigma\ :\ k\sigma \in \mathrm{pre}(v) \Rightarrow \pi(k\sigma) \in \delta(\sigma, \pi(k)).$$

Thus a path determines a state sequence consistent with the form of the string and the transition structure of the automaton.

Often, we wish to discuss "paths" that do not begin with the initial state (i.e., for which $\pi(1) \neq x_0$). For this, we define $\mathcal{A}_x$ to be the automaton obtained from $\mathcal{A}$ by designating $x \in X$ the initial state.

In keeping with our interpretation of automata as generators, we say that a string $v \in \Sigma^\infty$ is *generated* by $\mathcal{A}$ if v has a path on $\mathcal{A}$.

The *recurrence set* of a path $\pi : \mathrm{pre}(s) \longrightarrow X$ is

$$\Omega_\pi := \{x \in X : |\pi^{-1}(x)| = \omega\}.$$

In other words, the recurrence set is the set of states entered infinitely often on a given path. The recurrence set Ω_π is nonempty if and only if the string s is infinite.

For our purposes, it is convenient to adopt a slight modification of the standard definition of acceptance. A path π is *accepted* if

$$\exists p \in P : \Omega_\pi \cap R_p \neq \emptyset,\ \Omega_\pi \subseteq I_p.$$

Such paths represent infinite state sequences along which, for some $p \in P$, R_p is "continually recurrent" (is occupied infinitely often) and I_p is "eventually invariant" (is occupied almost always). (It is more customary to specify in place of I_p its complement, say $\overline{I_p}$, and require the equivalent condition that $\Omega_\pi \cap \overline{I_p} = \emptyset$, in other words, that $\overline{I_p}$ be "finitely recurrent" (occupied almost never).)

The ω-language *accepted* by $\mathcal{A}$ is the set of all (infinite) strings over Σ that have accepted paths on $\mathcal{A}$.

We say that $\mathcal{A}$ is *deterministic* if $|\delta(\sigma, x)| \leq 1$ for all $\sigma \in \Sigma, x \in X$. In this case, we represent the transition function as a partial function $\delta : \Sigma^* \times X \longrightarrow X$, writing $\delta(k, x)!$ to signify that the function is defined for the particular argument (k, x). If $\mathcal{A}$ is deterministic, then every string has at most one path π on $\mathcal{A}$.

According to our definition, a string is accepted by $\mathcal{A}$ if it has a path on $\mathcal{A}$ along which, for some $p \in P$, R_p is continually recurrent and I_p is eventually invariant. While this notion of acceptance may appear arbitrary, it is quite general in the sense that all of the languages that are accepted by finite ω-automata, the so-called ω-regular languages, are accepted by deterministic Rabin automata [26], [5], [9]. Indeed, the Rabin acceptance condition arises naturally in the "determinization" of nondeterministic ω-automata [26], [37], [14].

To introduce a control feature, we assume that a family $\mathbf{C} \subseteq 2^\Sigma$ of *control patterns* is given, representing subsets of the alphabet to which we can restrict, at any point

in the operation of the automaton, the set of symbols that it may generate. Control strategies can be represented as "feedback maps" $f : \Sigma^* \longrightarrow \mathbf{C}$ (pfn), which can be interpreted as associating with the sequence of all past symbols generated by the automaton a corresponding control action. Consistent with this interpretation, we say that $v \in \Sigma^\infty$ is generated by $\mathcal{A}$ *under* f if v is generated by $\mathcal{A}$, and, for all $k\sigma \in \text{pre}(v)$, $f(k)$ is defined and $\sigma \in f(k)$. We say that f is complete with respect to $\mathcal{A}$ if for all $k \in \Sigma^*$ generated by $\mathcal{A}$ under f, $f(k)$ is defined [34], [42].

We can now define the set from which the infinite behavior of an automaton can be controlled to the satisfaction of its acceptance condition. For any Rabin automaton $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$, define $P^{\mathcal{A}} \subseteq X$ as the set of all $x \in X$ for which there exists a complete feedback map $f : \Sigma^* \longrightarrow \mathbf{C}$ such that

1) Every $s \in \Sigma^\omega$ generated by $\mathcal{A}_x$ under f is accepted by $\mathcal{A}_x$, and

2) For any $k \in \Sigma^*$ generated by $\mathcal{A}_x$ under f, there exists $\sigma \in \Sigma$ such that $k\sigma$ is generated by $\mathcal{A}_x$ under f.

Clause 1) captures the notion that f should control $\mathcal{A}$ to the satisfaction of its acceptance condition. Clause 2) eliminates trivial solutions by requiring that every finite string generated by $\mathcal{A}_x$ under f have proper extensions that are also generated by $\mathcal{A}_x$ under f; for deterministic $\mathcal{A}$, this means that the control strategy represented by f must avoid system deadlocks.

We call $P^{\mathcal{A}}$ the *controllability subset* of $\mathcal{A}$. The above definition can be considered "operational" in the sense that it is stated in terms of the infinite strings generated by $\mathcal{A}$ under suitable control. While straightforward and intelligible from an intuitive standpoint, this description is of limited mathematical usefulness. The main results of the article establish an alternative representation that can be described as "denotational," in the sense that it is mathematically much more direct; it characterizes the controllability subset as a certain fixpoint of an operator that depends in a simple manner on the transition structure of the automaton and the family of control patterns. The new definition allows both efficient computation of the controllability subset and effective synthesis of appropriate controls.

4. Automaton structure. The recursive form of our fixpoint characterization of the controllability subset and the inductive nature of some of our proofs necessitate a precise notion of the structural complexity of automata and methods of converting complex automata to simpler ones. One useful measure of structural complexity is the number of pairs in the acceptance condition; another is the number of "live" states as defined by Rabin [33].

Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a Rabin automaton. The set of *live* states of $\mathcal{A}$ is given by[3]

$$L(\mathcal{A}) := \{x \in X : (\exists \sigma \in \Sigma)\ \delta(\sigma, x) \neq x\}.$$

In other words, a state is live if other states can be reached from it. An approximate opposite to liveness is "degeneracy." A state $x \in X$ is *degenerate* if there are transitions leaving x, but all of them simply lead back to x; more precisely, $x \in X$ is degenerate if

$$\exists \sigma \in \Sigma :\ \delta(\sigma, x)! \quad \forall \sigma \in \Sigma : \delta(\sigma, x) = x.$$

The set of degenerate states of $\mathcal{A}$ is denoted by $D(\mathcal{A})$. The subsets $L(\mathcal{A})$ and $D(\mathcal{A})$ are, of course, disjoint, but $L(\mathcal{A}) \cup D(\mathcal{A})$ may be a proper subset of X; indeed, $L(\mathcal{A}) \cup D(\mathcal{A}) = \{x \in X : (\text{there exists } \sigma \in \Sigma)\delta(\sigma, x)!\}$.

[3] Rabin excludes the initial state from the set of live states.

For any $x \in X$, $X' \subseteq X$, and $p \in P$, the following operations[4] on the Rabin automaton $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ potentially reduce its structural complexity as measured by $|L(\mathcal{A})|$ and $|P|$:

self-looping of a subset: $\mathcal{A}(\hookrightarrow X') := (\Sigma, X, \delta', x_0, \{(R'_p, I'_p) : p \in P\})$, where

$$\delta'(\sigma, x') = \begin{cases} x' & \text{if } x' \in X', \\ \delta(\sigma, x') & \text{otherwise,} \end{cases}$$
$$R'_p = R_p \cup X', \quad I'_p = I_p \cup X' \quad \forall p \in P;$$

restriction to a subset: $\mathcal{A} \restriction X' := (\Sigma, X, \delta', x_0, \{(R'_p, I'_p) : p \in P\})$, where

$$\delta'(\sigma, x') = \begin{cases} \delta(\sigma, x') & \text{if } x' \in X', \\ x' & \text{otherwise,} \end{cases}$$
$$R'_p = R_p \cap X', \quad I'_p = I_p \cap X' \quad \forall p \in P;$$

exclusion of a pair: Let $\mathcal{A} \restriction (I_p \cup D(\mathcal{A})) = (\Sigma, X, \delta', x_0, \{(R'_q, I'_q) : q \in P\})$. Then

$$\mathcal{A} \downarrow p := \begin{cases} (\Sigma, X, \delta', x_0, \{(R'_q, I'_q) : q \in P\}) & \text{if } |P| = 1, \\ (\Sigma, X, \delta', x_0, \{(R'_q, I'_q) : q \in P \setminus \{p\}\}) & \text{if } |P| > 1. \end{cases}$$

Self-looping of a subset $X' \subseteq X$ turns every $x \in X'$ into a degenerate state and ensures that the singleton $\{x\}$ satisfies the acceptance criterion. On the other hand, restriction to a subset $X' \subseteq X$ turns all other states into degenerate states that do *not* satisfy the acceptance condition. Finally, exclusion of a pair indexed by $p \in P$ restricts the automaton to the subset $I_p \cup D(\mathcal{A})$ and, provided that $|P| > 1$, eliminates the pair (R_p, I_p). All three of these operations potentially reduce the number of live states, while the third potentially reduces the number of pairs in the acceptance condition.

The effects of the operations on the subset $P^{\mathcal{A}}$ can be described as follows.

PROPOSITION 4.1. *Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a deterministic Rabin automaton and suppose that $x \in X$, $X' \subseteq X$ and $p \in P$. Then*

(a) $P^{\mathcal{A}} \cap D(\mathcal{A}) = \bigcup_{p \in P} (R_p \cap I_p) \cap D(\mathcal{A})$,

(b) $P^{\mathcal{A}} \cup X' \subseteq P^{\mathcal{A}(\hookrightarrow X')}$,

(c) $P^{\mathcal{A} \restriction X'} \subseteq P^{\mathcal{A}} \cap X'$,

(d) $P^{\mathcal{A} \downarrow p} \subseteq P^{\mathcal{A}} \cap (I_p \cup D(\mathcal{A}))$.

Proof. The proof follows by definition. □

Part (a) means intuitively that an automaton can be controlled to satisfy its acceptance condition from a degenerate state if and only if that degenerate state itself satisfies the acceptance criterion (since no other states can be reached from it). Part (b) states that self-looping makes it easier to force the automaton to satisfy its acceptance criterion by creating degenerate states that satisfy the criterion, while (c) and (d) state, respectively, that restriction and exclusion make it harder by creating degenerate states that do not satisfy the acceptance condition and by strengthening the acceptance condition.

5. The inverse dynamics operator and p-reachability operators. We characterize $P^{\mathcal{A}} \subseteq X$ as a certain fixpoint of the following monotone operator. Let

[4] The operations of "self-looping" and "restriction" are based on similar operations defined in [33].

$\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a deterministic Rabin automaton. Its *inverse dynamics operator* is given by

$$\theta^{\mathcal{A}} : 2^X \longrightarrow 2^X,$$
$$X' \mapsto \{x \in X : (\exists \Gamma \in \mathbf{C})[(\forall \sigma \in \Gamma)\delta(\sigma, x) \in X',\ (\exists \sigma \in \Gamma)\delta(\sigma, x)!]\}.$$

For any $X' \subseteq X$, $\theta^{\mathcal{A}}(X')$ is the set of all states in which the automaton can be controlled so that its next state belongs to X'.

The subset $P^{\mathcal{A}}$ is indeed one of the fixpoints of $\theta^{\mathcal{A}}$, as shown in the following result.

PROPOSITION 5.1. *Let $\mathcal{A}$ be a deterministic Rabin automaton. Then*

$$P^{\mathcal{A}} = \theta^{\mathcal{A}}(P^{\mathcal{A}}).$$

Proof. The proof follows by definition. □

This result does not determine $P^{\mathcal{A}}$ uniquely, since $\theta^{\mathcal{A}}$ may have many fixpoints, but we show that, with the use of the inverse dynamics operator, $P^{\mathcal{A}}$ can be uniquely represented by an expression in our fixpoint calculus. Other significant state subsets can be represented in similar fashion; for example, if $X_1 \subseteq X$, then

$$\nu X_2.\ [\theta^{\mathcal{A}}(X_2) \cap X_1]$$

denotes the supremal "control-invariant" subset of X_1 (see Theorem 2.1 (i′)); the fixpoint

$$\mu X_2.\ \theta^{\mathcal{A}}(X_1 \cup X_2)$$

is the "reachability subset" of X_1, the set of states from which the automaton can be controlled to reach X_1 (see Theorem 2.1(ii)); for $I_p \subseteq X$, the subset

$$\mu X_2.\ [\theta^{\mathcal{A}}(X_1 \cup X_2) \cap I_p]$$

is the set of states from which $\mathcal{A}$ can be controlled to reach $X_1 \subseteq X$ by way of a path that lies within[5] I_p.

To write a succinct expression for $P^{\mathcal{A}}$, it is convenient to generalize this second notion of reachability. Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a Rabin automaton. For any $p \in P$, the *p-reachability operator* of $\mathcal{A}$ is given by

$$\rho_p^{\mathcal{A}} : (2^X)^2 \longrightarrow 2^X,$$
$$(X_1, X_2) \mapsto \mu X_3.\ [\theta^{\mathcal{A}}(X_1) \cup [\theta^{\mathcal{A}}(X_1 \cup X_2 \cup X_3) \cap I_p]].$$

Thus $\rho_p^{\mathcal{A}}(\emptyset, X_2) \subseteq X$ is simply the set of states from which $\mathcal{A}$ can be controlled to reach X_2 by way of a path that lies within I_p. In general, $\rho_p^{\mathcal{A}}(X_1, X_2)$ is the set of states from which $\mathcal{A}$ can be controlled to reach X_2 by way of a path that lies within $I_p \subseteq X$ or, failing that, to reach X_1 (and to do so in at most one state transition after leaving I_p).

It is shown in the next section that the subset $P^{\mathcal{A}}$ can be described in terms of extremal fixpoints of the p-reachability operators. As preliminaries, we present the following two results, which respectively relate the inverse dynamics operator and the p-reachability operators to the structure of automata.

[5] With the possible exception of the final state in the path, which may belong to $X_1 \setminus I_p$.

PROPOSITION 5.2. *Let* $\mathcal{A} = (X, \Sigma, \delta, x_0, \{(R_p, I_p) : p \in P\})$ *be a Rabin automaton and suppose* $x \in X$, $p \in P$ *and* $X', X'' \subseteq X$. *Then*

(a) $\theta^{\mathcal{A}}(X') \cap D(\mathcal{A}) = X' \cap D(\mathcal{A})$,

(b) $\theta^{\mathcal{A}(\hookrightarrow X'')}(X') \setminus X'' = \theta^{\mathcal{A}}(X') \setminus X''$,

(c) $\theta^{\mathcal{A}\restriction X''}(X') \cap X'' = \theta^{\mathcal{A}}(X') \cap X''$,

(d) $\theta^{\mathcal{A}\restriction p}(X') \cap [I_p \cup D(A)] = \theta^{\mathcal{A}}(X') \cap [I_p \cup D(A)]$.

Part (a) simply means that all transitions leaving a degenerate state lead back to that state. Part (b) says that the self-looping of a subset X'' affects only transitions from states belonging to X''. On the other hand, part (c) says that restriction to $X'' \subseteq X$ affects only transitions from states *not* belonging to X''; part (d) is similar.

PROPOSITION 5.3. *Let* $\mathcal{A} = (X, \Sigma, \delta, x_0, \{(R_p, I_p) : p \in P\})$ *be a Rabin automaton and suppose that* $x \in X$, $X_1, X_2 \subseteq X$, *and* $p \in P$. *Then*

(a) $\rho_p^{\mathcal{A}}(X_1, X_2) \cap D(\mathcal{A}) = [X_1 \cup (X_2 \cap I_p)] \cap D(\mathcal{A})$,

(b) *If* $X' \subseteq X_1 \subseteq X$, $\rho_p^{\mathcal{A}(\hookrightarrow X')}(X_1, X_2) \setminus X' = \rho_p^{\mathcal{A}}(X_1, X_2) \setminus X'$,

(c) *If* $X_1 \subseteq X' \subseteq X$, $\rho_p^{\mathcal{A}\restriction X'}(X_1, X_2) \subseteq \rho_p^{\mathcal{A}}(X_1, X_2) \cap X'$,

(d) *If* $X_1, \rho_p^{\mathcal{A}}(X_1, X_2) \subseteq X'$ *and* $\rho_p^{\mathcal{A}\restriction X'}(X_1, X_2) = \rho_p^{\mathcal{A}}(X_1, X_2)$.

Proof. See [43]. □

Part (a) reflects the nature of degeneracy: degenerate states lead only to themselves. Part (b) states that the self-looping of a subset $X' \subseteq X_1$ does not affect the p-reachability of (X_1, X_2) from other states: intuitively, if the automaton ever reaches the subset X', then it will also have reached X_1, so further transitions (namely, those affected by the self-looping operation) will be irrelevant. Part (c) means that restriction to a subset $X' \supseteq X_1$ limits p-reachability of a pair (X_1, X_2) (by eliminating paths to X_1 that lie partly outside X' and by shrinking the set I_p). On the other hand, if the pair (X_1, X_2) is not p-reachable from X', then the restriction makes no difference; this is reflected in part (d).

6. Fixpoint characterization of the controllability subset. The p-reachability operators admit a concise representation of the controllability subset. Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a deterministic Rabin automaton. Then $C^{\mathcal{A}} \subseteq X$ is given by[6]

$$C^{\mathcal{A}} := \begin{cases} \mu X_1.\ \nu X_2.\ \rho_p^{\mathcal{A}}(X_1, X_2 \cap R_p)\,, & \text{if } |P| = 1, \\ \mu X_1.\ [\bigcup\limits_{p \in P} \nu X_2.\ \rho_p^{\mathcal{A}}(X_1, C^{\mathcal{A}(\hookrightarrow X_1 \cup (X_2 \cap R_p))\restriction p})] & \text{if } |P| > 1. \end{cases}$$

Consider the case where $|P| = 1$. By Theorem 2.1(ii), $C^{\mathcal{A}}$ is the least upper bound of the nondecreasing sequence of subsets $C_0 \subseteq C_1 \subseteq C_2 \subseteq \cdots$, given by

$$C_0 := \emptyset,$$
$$C_{i+1} := \nu X_2.\ \rho_p^{\mathcal{A}}(C_i, X_2 \cap R).$$

Intuitively, $C_1 \subseteq X$ is the largest $X_2 \subseteq X$ from which the automaton can be controlled to reach $X_2 \cap R_p$ by way of a path that lies within I_p (see Theorem 2.1(i′)); in other words, C_1 is the set of states from which the automaton can be controlled to remain forever within I_p and to enter R_p infinitely often. By induction, C_i represents the subset from which the automaton can be controlled so that it enters R_p infinitely often and enters $X \setminus I_p$ fewer than i times. Thus $C^{\mathcal{A}}$ is indeed the set of states from

[6] Existence of $C^{\mathcal{A}}$ follows by induction on $|P|$ from Proposition 6.2(b).

which $\mathcal{A}$ can be controlled to enter R_p infinitely often and eventually to enter I_p and remain there.

If $|P| > 1$, $C^{\mathcal{A}}$ is the least upper bound of the nondecreasing sequence $C_0 \subseteq C_1 \subseteq C_2 \subseteq \cdots$, given by

$$C_0 := \emptyset,$$
$$C_{i+1} := \bigcup_{p \in P} \nu X_2.\ \rho_p^{\mathcal{A}}(C_i, C^{\mathcal{A}(\hookrightarrow C_i \cup (X_2 \cap R_p))|p}).$$

Thus $C_1 \subseteq X$ is the set of states from which, for some $p \in P$, $\mathcal{A}$ can be controlled to remain forever within I_p and either enter R_p infinitely often or satisfy the acceptance condition obtained by excluding the pair (R_p, I_p), in other words, to remain forever within I_p and satisfy the original acceptance condition. By induction, C_i is the subset in which a sequence $p_i, p_{i-1}, p_{i-2}, \ldots, p_1$ of i elements of P can be inductively chosen in such a way that, if p is the latest element to have been selected, then $\mathcal{A}$ can be controlled so that it either remains within I_p forever and satisfies its acceptance condition or eventually reaches a state in which a new element in the sequence can be chosen. (When the last element p_1 is chosen, $\mathcal{A}$ is in a state from which it can be controlled to remain forever within I_{p_1} and satisfy its acceptance condition, that is, $\mathcal{A}$ is in C_1.) It follows that $C^{\mathcal{A}}$ is the subset from which $\mathcal{A}$ can be controlled to satisfy its acceptance condition.

As this interpretation suggests, $C^{\mathcal{A}}$ coincides with $P^{\mathcal{A}}$, as is shown in the next result.

PROPOSITION 6.1. *For any deterministic Rabin automaton $\mathcal{A}$, $P^{\mathcal{A}} = C^{\mathcal{A}}$.*

Proof. The proof follows by Proposition 6.3 and Theorem 6.4, below. □

Before proving the results that lead to Proposition 6.1, we establish some basic properties of $C^{\mathcal{A}}$ in Proposition 6.2.

PROPOSITION 6.2. *Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a deterministic Rabin automaton. Suppose that $x \in X$, $X' \subseteq X$, and $p \in P$. Then*

(a) $C^{\mathcal{A}} \cap D(\mathcal{A}) = \bigcup_{p \in P}(R_p \cap I_p) \cap D(\mathcal{A})$,

(b) $C^{\mathcal{A}(\hookrightarrow X')} \supseteq C^{\mathcal{A}} \cup X'$,

(c) $C^{\mathcal{A}(\hookrightarrow X')} = C^{\mathcal{A}} \iff X' \subseteq C^{\mathcal{A}}$,

(d) $C^{\mathcal{A}|X'} \subseteq C^{\mathcal{A}} \cap X'$,

(e) $C^{\mathcal{A}|p} \subseteq C^{\mathcal{A}} \cap [I_p \cup D(\mathcal{A})]$,

(f) $C^{\mathcal{A}} = \theta^{\mathcal{A}}(C^{\mathcal{A}})$,

(g) *if* $L(\mathcal{A}) \subseteq I_p$ *and* $X' \supseteq C^{\mathcal{A}(\hookrightarrow X' \cap R_p \cap I_p)} \cap R_p \cap I_p$ *then* $C^{\mathcal{A}(\hookrightarrow X' \cap R_p \cap I_p)|p} = C^{\mathcal{A}(\hookrightarrow X' \cap R_p \cap I_p)}$,

(h) *if* $L(\mathcal{A}) \subseteq I_p$ *then* $C^{\mathcal{A}} = \nu X_2.\ \theta^{\mathcal{A}}(C^{\mathcal{A}(\hookrightarrow X_2 \cap R_p \cap I_p)})$.

Proof. See [43]. □

Part (a) means that the automaton can be controlled to satisfy its acceptance condition from a degenerate state if and only if that degenerate state itself satisfies the acceptance criterion. This is natural, since degenerate states lead only to themselves.

Part (b) describes the way the controllability subset generally expands when a subset is self-looped, owing to the creation of degenerate states that satisfy the acceptance criterion. Part (c) states that the controllability subset is unaffected by this operation if and only if the states that are self-looped already belong to the controllability subset.

Parts (d) and (e) reflect the fact that restriction to a subset and exclusion of a pair shrink the controllability subset by creating degenerate states that do not satisfy the acceptance criterion and by strengthening the acceptance criterion.

Part (f) means simply that the controllability subset is a fixpoint of the inverse dynamics operator.

Part (g) describes a situation in which exclusion of a pair (R_p, I_p) does not affect the controllability subset, namely, that in which all live states belong to I_p (so that the restriction to $I_p \cup D(\mathcal{A})$ is of no consequence) and a sufficiently large subset of R_p has been self-looped (so that the elimination of the pair (R_p, I_p) does not strengthen the acceptance condition).

Finally, part (h) states that, when all live states belong to some I_p, the controllability subset is the set of all states from which the automaton can be controlled to satisfy its acceptance condition or simply to enter $R_p \cap I_p$ infinitely often (the invariance of I_p following automatically).

These preliminary results allow us to prove that $P^{\mathcal{A}}$ and $C^{\mathcal{A}}$ coincide (Proposition 6.1). We first establish that $P^{\mathcal{A}} \subseteq C^{\mathcal{A}}$; in other words, if $\mathcal{A}$ can be controlled to satisfy its acceptance condition from state x, then $x \in C^{\mathcal{A}}$:

PROPOSITION 6.3. *For any deterministic Rabin automaton $\mathcal{A}$, $P^{\mathcal{A}} \subseteq C^{\mathcal{A}}$.*

Proof. We proceed by induction on the number of live states of $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$, in the manner of [33]. Note that

$$\begin{aligned} & P^{\mathcal{A}} \cap D(\mathcal{A}) \\ &= \bigcup_{p \in P} (R_p \cap I_p) \cap D(\mathcal{A}) \qquad \text{(Prop. 4.1(a))} \\ &= C^{\mathcal{A}} \cap D(\mathcal{A}) \qquad \text{(Prop. 6.2(a))}. \end{aligned}$$

By Proposition 5.1, it thus suffices to show that $P^{\mathcal{A}} \cap L(\mathcal{A}) \subseteq C^{\mathcal{A}}$.

If $\mathcal{A}$ contains no live states, then the result holds vacuously. For the induction step, suppose that $x \in P^{\mathcal{A}} \cap L(\mathcal{A})$ and assume that the result holds for all Rabin automata with fewer live states than $\mathcal{A}$. We prove that $x \in C^{\mathcal{A}}$.

By assumption, there exists some complete feedback map $f : \Sigma^* \longrightarrow \mathbf{C}$ satisfying both clauses of the definition of $P^{\mathcal{A}}$. The central issue in the proof is the nature of the relationship between the strings generated by $\mathcal{A}_x$ under f and the live states of $\mathcal{A} =: (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$. The following three cases exhaust the possibilities:

(a) There exists a live state $x' \in X$ such that, for all $k' \in \Sigma^*$ generated by $\mathcal{A}_x$ under f, $\delta(k', x) \neq x'$;

(b) For some pair of live states $x', x'' \in X$, there exists $k' \in \Sigma^*$ generated by $\mathcal{A}_x$ under f such that $\delta(k', x) = x'$ and for all $k'' \in \Sigma^*$ generated by $\mathcal{A}_x$ under f such that $k' < k''$, $\delta(k'', x) \neq x''$;

(c) For all pairs of live states $x', x'' \in X$, and every $k' \in \Sigma^*$ generated by $\mathcal{A}_x$ under f such that $\delta(k', x) = x'$, there exists $k'' \in \Sigma^*$ generated by $\mathcal{A}_x$ under f such that $k' < k''$ and $\delta(k'', x) = x''$.

In case (a), we have

$$\begin{aligned} x &\in P^{\mathcal{A}\restriction(X \setminus \{x'\})} \\ &\subseteq C^{\mathcal{A}\restriction(X \setminus \{x'\})} \qquad \text{(by inductive hypothesis)} \\ &\subseteq C^{\mathcal{A}} \quad \text{(Prop. 6.2(d))}. \end{aligned}$$

Similarly, for case (b), we have

$$\begin{aligned} x' = \delta(k', x) &\in P^{\mathcal{A}\restriction(X \setminus \{x''\})} \\ &\subseteq C^{\mathcal{A}\restriction(X \setminus \{x''\})} \qquad \text{(by inductive hypothesis)} \\ &\subseteq C^{\mathcal{A}} \qquad \text{(Prop. 6.2(d))}. \end{aligned}$$

Thus

$$\begin{aligned} x &\in P^{\mathcal{A}} \\ &\subseteq P^{\mathcal{A}(\hookrightarrow x')} && \text{(Prop. 4.1(b))} \\ &\subseteq C^{\mathcal{A}(\hookrightarrow x')} && \text{(by inductive hypothesis)} \\ &= C^{\mathcal{A}} && \text{(Prop. 6.2(c)).} \end{aligned}$$

In case (c), there exists a string s generated by $\mathcal{A}_x$ under f having a path π on $\mathcal{A}$ such that $\Omega_\pi = L(\mathcal{A})$. It follows that, for some $p \in P$,

$$L(\mathcal{A}) \subseteq I_p \quad \text{and} \quad L(\mathcal{A}) \cap R_p \neq \emptyset.$$

Furthermore, for any $x''' \in L(\mathcal{A})$, we have

$$\begin{aligned} x''' &\in P^{\mathcal{A}} \\ &= \theta^{\mathcal{A}}(P^{\mathcal{A}}) && \text{(Prop. 5.1)} \\ &\subseteq \theta^{\mathcal{A}}(P^{\mathcal{A}(\hookrightarrow L(\mathcal{A})\cap R_p)}) && \text{(Prop. 4.1 (b))} \\ &= \theta^{\mathcal{A}}(C^{\mathcal{A}(\hookrightarrow L(\mathcal{A})\cap R_p)}) && \text{(by inductive hypothesis).} \end{aligned}$$

Thus

$$\begin{aligned} L(\mathcal{A}) &\subseteq \nu X_2.\ \theta^{\mathcal{A}}(C^{\mathcal{A}(\hookrightarrow X_2 \cap R_p \cap I_p)}) && \text{(Thm. 2.1(i'))} \\ &= C^{\mathcal{A}} && \text{(Prop. 6.2(h))} \end{aligned}$$

This completes the induction. □

The next result establishes the converse of Proposition 6.3, namely, it shows that, if the deterministic Rabin automaton $\mathcal{A}$ is in state $x \in C^{\mathcal{A}}$, then it can be controlled to satisfy its acceptance condition. Moreover, it states that "state feedback" is sufficient. Together, Proposition 6.3 and Theorem 6.4 thus imply that a Rabin automaton can be controlled to satisfy its acceptance condition if and only if it can be so controlled by means of state feedback.

THEOREM 6.4 (state feedback). *Let $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ be a deterministic Rabin automaton. Then there exists a total map $\phi^{\mathcal{A}} : C^{\mathcal{A}} \longrightarrow \mathbf{C}$ such that*

1) *For any $s \in \Sigma^\omega$ and any path $\pi : \mathrm{pre}(s) \longrightarrow X$ of s on $\mathcal{A}$, the condition*

$$\pi(1) \in C^{\mathcal{A}} \quad \text{and} \quad \forall k\sigma \in \mathrm{pre}(s) \ :\ \sigma \in \phi^{\mathcal{A}}(\pi(k))$$

implies that

$$\exists p \in P \ :\ \Omega_\pi \cap R_p \neq \emptyset \quad \text{and} \quad \Omega_\pi \subseteq I_p;$$

2) *For any $x \in C^{\mathcal{A}}$,*

$$\begin{gathered} \exists \sigma \in \phi^{\mathcal{A}}(x) \ :\ \delta(\sigma, x)!, \\ \forall \sigma \in \phi^{\mathcal{A}}(x) \ :\ \delta(\sigma, x)! \implies \delta(\sigma, x) \in C^{\mathcal{A}}. \end{gathered}$$

Proof. The details of the proof are intricate, but the methods are simple. We consider every least fixpoint as the limit of a nondecreasing sequence of state subsets, by Theorem 2.1. Membership in these subsets is used to define a well-founded partial ordering or "ranking" of states: the earlier a state occurs in the sequence of subsets, the lower its rank. The theorem is then proved through arguments concerning

the effect of a suitable state feedback control on various ranks of the system state; these typically show that, under appropriate conditions, a particular rank is either nonincreasing or strictly decreasing.

We construct a suitable map by induction on $|P|$. We give only the induction step; the base of the induction (dealing with the case where $|P| = 1$) is similar; see [43] for details.

Suppose then that $|P| > 1$ and assume that the result holds for all automata having fewer pairs. Then

$$C^{\mathcal{A}} = \mu X_1.\ \left[\bigcup_{p \in P} \nu X_2.\ \rho_p^{\mathcal{A}}(X_1, C^{\mathcal{A}(\hookrightarrow X_1 \cup (X_2 \cap R_p))\downarrow p}) \right].$$

By Theorem 2.1, this fixpoint is the least upper bound of the nondecreasing sequence $C_0^{\mathcal{A}} \subseteq C_1^{\mathcal{A}} \subseteq C_2^{\mathcal{A}} \subseteq \cdots$, given by

$$\begin{aligned} C_0^{\mathcal{A}} &:= \emptyset, \\ C_{i+1}^{\mathcal{A}} &:= \textstyle\bigcup_{p \in P} \nu X_2.\ \rho_p^{\mathcal{A}}(C_i^{\mathcal{A}}, C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (X_2 \cap R_p))\downarrow p}) \\ &= \textstyle\bigcup_{p \in P} C_{i+1,p}^{\mathcal{A}}, \end{aligned}$$

where

$$\begin{aligned} C_{i+1,p}^{\mathcal{A}} &:= \nu X_2.\ \rho_p^{\mathcal{A}}(C_i^{\mathcal{A}}, C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (X_2 \cap R_p))\downarrow p}) \\ &= \rho_p^{\mathcal{A}}(C_i^{\mathcal{A}}, C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C_{i+1,p}^{\mathcal{A}} \cap R_p))\downarrow p}) \\ &= \mu X_3.\ \ [\theta^{\mathcal{A}}(C_i^{\mathcal{A}}) \\ &\qquad \cup [\theta^{\mathcal{A}}(C_i^{\mathcal{A}} \cup C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C_{i+1,p}^{\mathcal{A}} \cap R_p))\downarrow p} \cup X_3) \cap I_p]]. \end{aligned}$$

Each $C_{i+1,p}^{\mathcal{A}}$ is the least upper bound of the nondecreasing sequence, given by

$$\begin{aligned} C_{i+1,p,0}^{\mathcal{A}} &:= \emptyset, \\ C_{i+1,p,j+1}^{\mathcal{A}} &:= [\theta^{\mathcal{A}}(C_i^{\mathcal{A}}) \\ &\qquad \cup [\theta^{\mathcal{A}}(C_i^{\mathcal{A}} \cup C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C_{i+1,p}^{\mathcal{A}} \cap R_p))\downarrow p} \cup C_{i+1,p,j}^{\mathcal{A}}) \cap I_p]] \\ &= C_{i+1,p,j+1,0}^{\mathcal{A}} \cup C_{i+1,p,j+1,1}^{\mathcal{A}}, \end{aligned}$$

where

$$\begin{aligned} C_{i+1,p,j+1,0}^{\mathcal{A}} &:= \theta^{\mathcal{A}}(C_i^{\mathcal{A}}), \\ C_{i+1,p,j+1,1}^{\mathcal{A}} &:= \theta^{\mathcal{A}}(C_i^{\mathcal{A}} \cup C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C_{i+1,p}^{\mathcal{A}} \cap R_p))\downarrow p} \cup C_{i+1,p,j}^{\mathcal{A}}) \cap I_p. \end{aligned}$$

Thus

$$C^{\mathcal{A}} = \bigcup_{i=1}^{|X|} \bigcup_{p \in P} \bigcup_{j=1}^{|X|} \bigcup_{k=0,1} C_{i,p,j,k}^{\mathcal{A}}.$$

Define total feedback maps $\phi_{i,p,j,k}^{\mathcal{A}} : C_{i,p,j,k}^{\mathcal{A}} \longrightarrow \mathbf{C}$ for each of the components $C_{i,p,j,k}^{\mathcal{A}}$ as follows:

- If $k = 0$, define $\phi_{i+1,p,j+1,k}^{\mathcal{A}} : C_{i+1,p,j+1,k}^{\mathcal{A}} \longrightarrow \mathbf{C}$ so that, for all $x \in C_{i+1,p,j+1,0}^{\mathcal{A}}$,

$$\forall \sigma \in \phi_{i+1,p,j+1,k}^{\mathcal{A}}(x)\ :\ \delta(\sigma, x) \in C_i^{\mathcal{A}};$$

• If $k = 1$, define $\phi^{\mathcal{A}}_{i+1,p,j+1,k} : C^{\mathcal{A}}_{i+1,p,j+1,k} \longrightarrow \mathbf{C}$ so that, for all $x \in C^{\mathcal{A}}_{i+1,p,j+1,1} \subseteq I_p$,

$$\forall \sigma \in \phi^{\mathcal{A}}_{i+1,p,j+1,k}(x) : \\ \delta(\sigma, x) \in C^{\mathcal{A}}_i \cup C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p} \cup C^{\mathcal{A}}_{i+1,p,j} .$$

Choose a total ordering of P and order the 4-tuples (i,p,j,k) lexicographically. Define a total map $\phi^{\mathcal{A}} : C^{\mathcal{A}} \longrightarrow \mathbf{C}$ so that

$$\phi^{\mathcal{A}} : x \mapsto \begin{cases} \phi^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}(x) & \text{if } x \in C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p} \setminus C^{\mathcal{A}}_i \cup R_p, \\ \phi^{\mathcal{A}}_{i+1,p,j+1,k}(x) & \text{otherwise,} \end{cases}$$

where (i,p,j,k) is the least 4-tuple in the lexicographic ordering such that $x \in C^{\mathcal{A}}_{i+1,p,j+1,k}$.

Clause 2) of the theorem follows from the definition of $\phi^{\mathcal{A}}$ (by the inductive hypothesis). For clause 1), suppose that $s \in \Sigma^\omega$ and there exists a path $\pi : \mathrm{pre}(s) \longrightarrow X$ of s on $\mathcal{A}$ such that

$$\pi(1) = x \quad \text{and} \quad (\forall k\sigma \in \mathrm{pre}(s))[\sigma \in \phi^{\mathcal{A}}(\pi(k))].$$

We must show that there exists $p \in P \ : \ \Omega_\pi \cap R_p \neq \emptyset$ and $\Omega_\pi \subseteq I_p$.

Define

$$I\text{−rank} : C^{\mathcal{A}} \longrightarrow \mathbb{N} \times P, \\ x \mapsto (i,p),$$

where (i,p) is the least pair (in the lexicographic ordering) such that $x \in C^{\mathcal{A}}_{i,p}$.

Note that

$$\begin{aligned} & C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p} \setminus (C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p) \cup D(\mathcal{A})) \\ & = \theta^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}(C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}) \setminus (C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p) \cup D(\mathcal{A})) \\ & \qquad \text{(Prop. 6.2 (f))} \\ & = [\theta^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}(C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}) \cap I_p] \setminus (C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p) \\ & \qquad \cup D(\mathcal{A})) \text{ (Prop. 6.2 (e))} \\ & = [\theta^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))}(C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}) \cap I_p] \setminus (C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p) \\ & \qquad \cup D(\mathcal{A})) \text{ (Prop. 5.2 (d))} \\ & = [\theta^{\mathcal{A}}(C^{\mathcal{A}(\hookrightarrow C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\restriction p}) \cap I_p] \setminus (C^{\mathcal{A}}_i \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p) \cup D(\mathcal{A})) \\ & \qquad \text{(Prop. 5.2 (b))} \\ & \subseteq C^{\mathcal{A}}_{i+1,p}. \end{aligned}$$

It follows, by the definition of $\phi^{\mathcal{A}}$ and the inductive hypothesis, that, for all $x \in C^{\mathcal{A}}, \sigma \in \phi^{\mathcal{A}}(x)$, if $\delta(\sigma,x) \notin D(\mathcal{A})$, then

$$I\text{−rank}(\delta(\sigma,x)) \leq I\text{−rank}(x),$$

and, if $I\text{−rank}(x) = (i,p)$ and $x \notin I_p \cup D(\mathcal{A})$, then, by Proposition 6.2(e),

$$I\text{−rank}(\delta(\sigma,x)) < I\text{−rank}(x) \quad \forall \sigma \in \phi^{\mathcal{A}}(x).$$

If (i,p) is the least pair for which $\Omega_\pi \cap C^{\mathcal{A}}_{i+1,p} \neq \emptyset$, then it follows that $\Omega_\pi \subseteq C^{\mathcal{A}}_{i+1,p} \cap (I_p \cup D(\mathcal{A}))$. Note that, if $\Omega_\pi \cap C^{\mathcal{A}}_{i+1,p} \cap D(\mathcal{A}) \neq \emptyset$, then there exists $q \in P \ : \ \Omega_\pi \cap R_q \neq \emptyset$

and $\Omega_\pi \subseteq I_q$ holds (by Proposition 6.2(a)). We may therefore assume instead that $\Omega_\pi \subseteq I_p$.

Define

$$R\text{-rank} : C^{\mathcal{A}} \longrightarrow \mathbb{N} \times P \times \mathbb{N},$$
$$x \mapsto (i, p, j)$$

where (i, p, j) is the least triple such that $x \in C^{\mathcal{A}}_{i,p,j}$.

By definition of $\phi^{\mathcal{A}}$, if $x \in C^{\mathcal{A}}$, $\sigma \in \phi^{\mathcal{A}}(x)$, $I\text{-rank}(x) = (i+1, p)$, and $\delta(\sigma, x) \notin C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\downarrow p}$, then

$$R\text{-rank}(\delta(\sigma, x)) < R\text{-rank}(x).$$

If (i, p) is the least pair such that $\Omega_\pi \cap C^{\mathcal{A}}_{i+1,p} \neq \emptyset$, then it follows that $\Omega_\pi \cap C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\downarrow p} \neq \emptyset$. Then, however, by the inductive assumption, we have either $\Omega_\pi \cap R_p \neq \emptyset$ or $\Omega_\pi \subseteq C^{\mathcal{A}(\hookrightarrow C_i^{\mathcal{A}} \cup (C^{\mathcal{A}}_{i+1,p} \cap R_p))\downarrow p} \setminus (C_i^{\mathcal{A}} \cup R_p)$. The result follows by another application of the inductive assumption. □

7. The complexity of computing controllability subsets. In this section, we demonstrate that the computation of controllability subsets by straightforward calculation of the appropriate fixpoint is essentially optimal. We first establish that the problem is NP-complete and then show that calculation of the fixpoint is singly exponential in the number of accepting pairs and polynomial in the size of the state set. These results match the strongest known results for the polynomially equivalent emptiness problem for Rabin automata on infinite trees [13], [29].

THEOREM 7.1. *The problem of deciding membership in the controllability subset of a deterministic Rabin automaton is* NP*-complete.*

Proof. NP*-hardness.* The proof follows by reduction from the emptiness problem for Rabin automata on infinite trees over a one-symbol alphabet. This was shown to be NP-hard in [13].

Membership in NP. If a Rabin automaton $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ is deterministic, then it follows from Proposition 6.3 and Theorem 6.4 that $x \in C^{\mathcal{A}}$ if and only if there exists a "state feedback" map $\phi : X \longrightarrow \mathbf{C}$ (pfn) such that $x \in \text{dom}(\phi)$ and if the following conditions hold:

1) For any $s \in \Sigma^\omega$ and any path $\pi : \text{pre}(s) \longrightarrow X$ of s on $\mathcal{A}$, the condition

$$\pi(1) \in \text{dom}(\phi) \quad \text{and} \quad \forall k\sigma \in \text{pre}(s) \ : \ \sigma \in \phi^{\mathcal{A}}(\pi(k))$$

implies that

$$\exists p \in P \ : \ \Omega_\pi \cap R_p \neq \emptyset \quad \text{and} \quad \Omega_\pi \subseteq I_p;$$

2) For any $x' \in \text{dom}(\phi)$,

$$\exists \sigma \in \phi(x') \ : \ \delta(\sigma, x')! \text{ and}$$
$$\forall \sigma \in \phi(x') \ : \ \delta(\sigma, x')! \implies \delta(\sigma, x') \in \text{dom}(\phi).$$

However, any map $\phi : X \longrightarrow \mathbf{C}$ can be constructed in polynomial time, and the conditions 1) and 2) can be checked in polynomial time, using the results of [15]. NP-hardness follows. See [43] for details. □

THEOREM 7.2. *The controllability subset of a deterministic Rabin automaton $\mathcal{A}$ can be computed in time $\bigcirc(kl(mn)^{3m})$, where k is the number of control patterns,*

l is the size of the alphabet, m is the number of state subset pairs in the acceptance condition, and n is the number of states.

Proof. The proof follows by straightforward analysis of the fixpoint computation. See [43] for details. □

8. Example. Consider the Rabin automaton $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ pictured in Fig. 8.1; the index set P is $\{\alpha, \beta\}$ and the pairs $(R_\alpha, I_\alpha) = (\{4\}, \{1, 2, 3, 4\})$ and $(R_\beta, I_\beta) = (\{-4\}, \{-1, -2, -3, -4\})$ are represented by the pairs of dotted and dashed boxes.

For simplicity, we take the alphabet Σ to be X^2; the transition function is represented by the arcs of the diagram, the symbol associated with a transition from state i to j being (i, j). The automaton is thus deterministic.

The family $\mathbf{C}$ of control patterns is the set of all subsets of Σ that contain the following symbols:

$$(0, \pm 1),\ (\pm 1, \mp 1),\ (\pm 3, \pm 4),\ (\pm 4, \pm 3),\ (\pm 2, \pm 3),\ (\pm 2, \pm 4).$$

This means that the state transitions corresponding to these symbols cannot be prevented, while all others can.

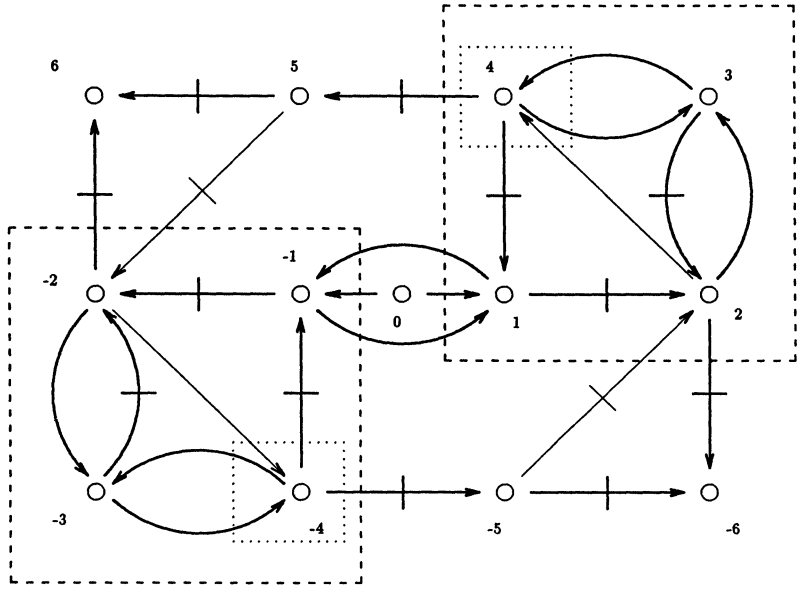


FIG. 8.1. *Rabin automaton $\mathcal{A}$.*

The controllability subset is computed by recursive application of (ii) and (ii′) of Theorem 2.1. The calculation is displayed in Table 8.1 using the notation of [43].

At the beginning of the procedure and after completing the first column of the table, it is necessary to construct an automaton of the form $\mathcal{A}(\hookrightarrow X_1 \cup (X_2 \cap R_p)) \downarrow p$ for every $p \in P$ and to compute its controllability subset. In our example, we begin the computation by constructing the automaton

$$\mathcal{A}(\hookrightarrow \emptyset \cup (X \cap R_\alpha)) \downarrow \alpha = \mathcal{A}(\hookrightarrow R_\alpha) \downarrow \alpha,$$

shown in Fig. 8.2, and calculating $C^{\mathcal{A}(\hookrightarrow R_\alpha)\downarrow\alpha}$, as shown in Table 8.2. (The corresponding results for $\beta \in P$ are obtained by replacing the states with their negatives.)

TABLE 8.1
Computation of $C^{\mathcal{A}}$.

	$C^{\mathcal{A}}_{1,\alpha,1,1}$	$C^{\mathcal{A}}_{2,\alpha,1,0}$	$\phi^{\mathcal{A}}(x)$
6			
5		α, β	$\Sigma \setminus \{(5,6)\}$
4	α	α, β	$\Sigma \setminus \{(4,1),(4,5)\}$
3	α	α, β	$[\Sigma \setminus \{(3,2)\}]$
2	α	α, β	$[\Sigma \setminus \{(2,-6)\}]$
1			
0			
-1			
-2	β	α, β	$[\Sigma \setminus \{(-2,6)\}]$
-3	β	α, β	$[\Sigma \setminus \{(-3,-2)\}]$
-4	β	α, β	$\Sigma \setminus \{(-4,-1),(-4,-5)\}$
-5		α, β	$\Sigma \setminus \{(-5,-6)\}$
-6			

$C^{\mathcal{A}}_{1,\beta,1,1} = C^{\mathcal{A}}_{2,\beta,1,0}$

Once the first column of Table 8.1 is computed, we construct

$$\mathcal{A}(\hookrightarrow X_1 \cup (X \cap R_\alpha)) \downarrow \alpha = \mathcal{A}(\hookrightarrow X_1) \downarrow \alpha$$

(where X_1 is the set of states marked by either α or β in the first column of Table 8.1) and calculate its controllability subset. The results are displayed in Fig. 8.3 and Table 8.3. (The corresponding results for β are again obtained by symmetry.)

Note that the states ± 6 are excluded from $C^{\mathcal{A}}$; they are "dead ends" from which no other state can be reached. The states $-1, 0, 1$ are excluded because of the cycle formed by 1 and -1.

A state feedback controller is given by the last column of Table 8.1. It is computed by identifying the subsets defined in the proof of Theorem 6.4 with the appropriate entries in the table and assigning control patterns that satisfy the rules set out in the proof. Those control patterns that appear in square brackets are taken from the feedback maps for $\mathcal{A}(\hookrightarrow R_\alpha) \downarrow \alpha$ and $\mathcal{A}(\hookrightarrow R_\beta) \downarrow \beta$, in accordance with the rules. The transition structure of Fig. 8.4 represents the set of strings generated by the automaton under the state feedback map. It can be plainly seen that every infinite string generated is accepted by $\mathcal{A}$ and that every finite string generated has an extension that is also generated under the state feedback control.

9. Discussion. We have presented a procedure for the computation of the *controllability subset* of a deterministic Rabin automaton, namely, the set of states from which the automaton can be controlled to the satisfaction of its acceptance condition. The key to the method is the representation of the controllability subset as a fixpoint of an *inverse dynamics operator*, which depends only on the one-step dynamics of the controlled system. Straightforward computation of this fixpoint matches tight upper bounds on the complexity of the problem. Moreover, intermediate results of the calculation allow the construction of a state feedback map that provides suitable control action.

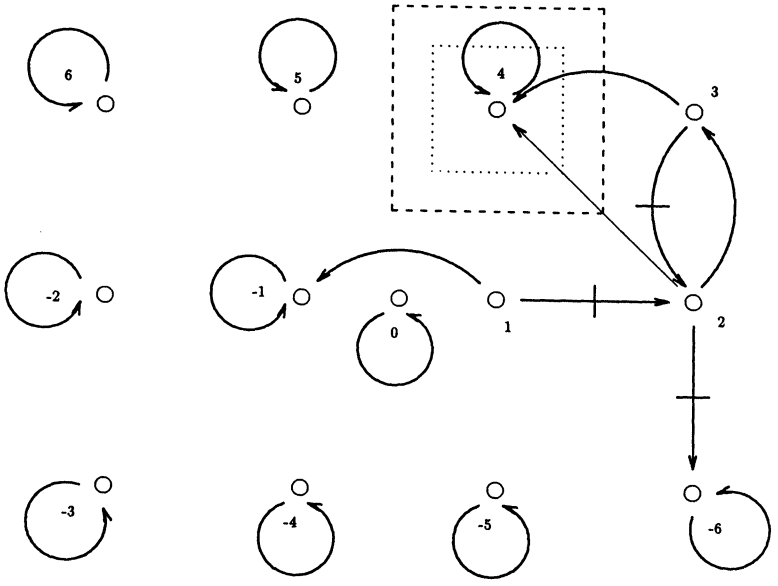


FIG. 8.2. *Simplified automaton* $\mathcal{A}(\hookrightarrow R_\alpha) \downarrow \alpha$.

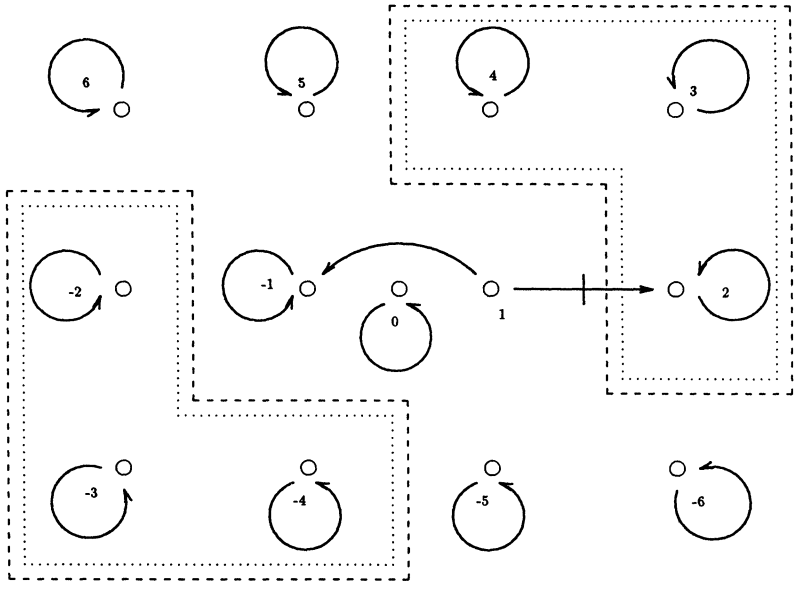


FIG. 8.3. *Simplified automaton* $\mathcal{A}(\hookrightarrow X_1) \downarrow \alpha$, *where* $X_1 = \{-4, -3, -2, 2, 3, 4\}$. (*The dotted and dashed boxes represent a single accepting pair.*)

The problem studied in the article was, in effect, first solved by Büchi and Landweber [24], [4], who applied game-theoretic techniques to the study of Church's problem [7]. Simpler solutions were later obtained through the equivalent emptiness problem for automata on infinite trees [33], [29]. The earliest approaches to the emptiness problem were developed by Rabin [33] and Hossley and Rackoff [20], employing, respectively, the structural induction method used in the present article and a reduction to the emptiness problem for automata on finite trees.

TABLE 8.2
Computation of $C^{\mathcal{A}(\hookrightarrow R_\alpha)\downarrow\alpha}$.

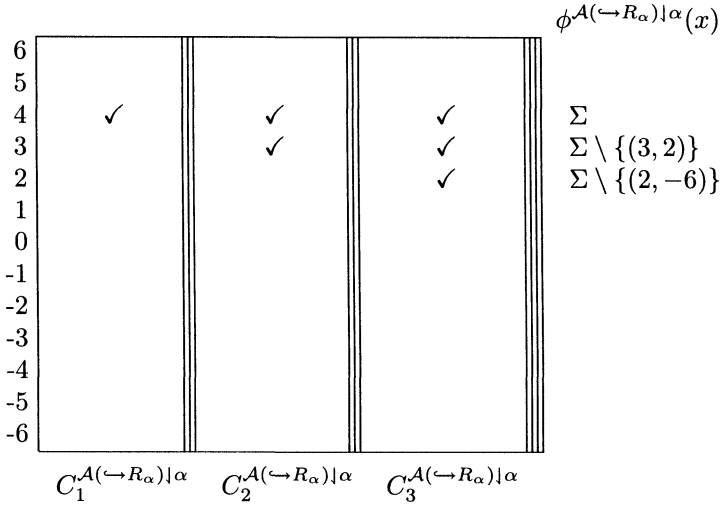
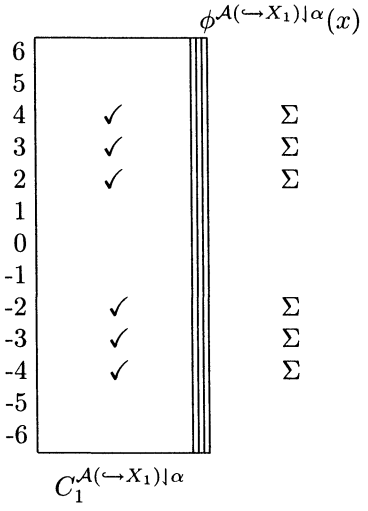


TABLE 8.3
Computation of $C^{\mathcal{A}(\hookrightarrow X_1)\downarrow\alpha}$, *where* $X_1 = \{-4, -3, -2, 2, 3, 4\}$.



A key result of both of these approaches is the so-called "finite model theorem," which states roughly that a given tree automaton accepts some infinite tree if and only if it accepts an infinite tree that has a finite description; moreover, such a tree can be effectively constructed. This result was strengthened by Emerson to a "small model theorem," which states that an automaton accepts some infinite tree if and only if it accepts an infinite tree obtained by "unwinding" some finite graph embedded in its own transition structure [11]. Emerson's proof is not directly constructive.[7] The state

[7] The small model theorem does not hold for more general *Muller* automata, but Gurevich and Harrington have established a positive result stating roughly that a Muller automaton can be controlled to satisfy its own acceptance condition if and only if it can be so controlled by means of

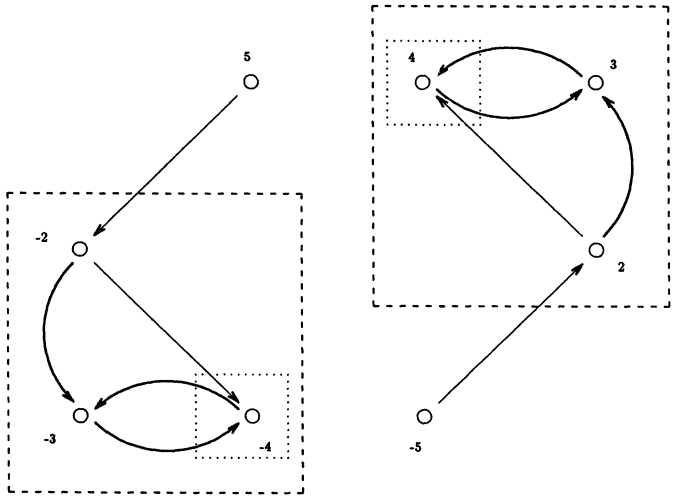


FIG. 8.4. *Transition structure representing strings generated by $\mathcal{A}$ under state feedback.*

feedback result of the current article (Theorem 6.4) is a new, constructive version of the small model theorem.

Two recent approaches to the emptiness problem are those of Pnueli and Rosner [29] and Emerson and Jutla [13]. These have established the tight upper bound on the computational complexity of the problem given in §7. Pnueli and Rosner's method was to refine the technique of Hossley and Rackoff, while Emerson and Jutla applied results on logical model-checking, expressing the acceptance condition in temporal logic, translating the resulting formula into a logical fixpoint calculus, and then checking for the existence of a model of the resulting formula embedded in the automaton's transition structure. The efficacy of this approach follows from the small model theorem.

In technical terms, the method of this paper is essentially a synthesis of those of Rabin and Emerson and Jutla, employing Rabin's method of induction on the number of "live" states and Emerson and Jutla's use of a fixpoint calculus and induction on the number of accepting pairs. The new solution is computationally simpler than Rabin's and mathematically more direct than Emerson and Jutla's: unlike those of [13], the results of this article do not depend on the small model theorem; in fact, the small model theorem is essentially a corollary of the main results of this report.

In addition to these technical advantages, the current approach has a system-theoretic flavor, which, in our opinion, renders it more transparent than the combinatorial, automata-theoretic techniques of [33], [20], [29] or the model-theoretic methods of [13]. This, together with its mathematical directness, suggests that the new technique more readily admits useful extensions. Indeed, in [41] the authors outline a generalization of the present results that allows for liveness assumptions represented by Büchi acceptance conditions. This solves an instance of a problem posed, but not constructively solved, in [1]. The methods of [20], [29] do not appear to admit such an extension.[8]

feedback of its own state and that of a buffer that stores the sequence of the most recent visits to the respective accepting subsets [19], [50].

[8] The treatment of liveness in Wong-Toi and Dill [49] (i.e., as a qualification of the specification) is inappropriate for our setting.

To conclude, this paper presents a direct, efficient, and natural solution to a basic problem in discrete-event system theory having applications to control synthesis, program synthesis, and logical decidability. The results illustrate fruitful interchange between control and computer science.

Acknowledgments. The reading of a preliminary version of these results by the first author's thesis examiners, particularly Professors Eric Hehner, Raymond Kwong, and Amir Pnueli, is gratefully acknowledged.

REFERENCES

[1] M. ABADI, L. LAMPORT, AND P. WOLPER, *Realizable and unrealizable specifications of reactive systems*, in Automata, Languages and Programming, 16th Internat. Colloquium, Stresa, Italy, July 1989, Proceedings (Lecture Notes in Computer Science, No. 372), Springer-Verlag, Berlin, New York, 1989, pp. 1–17.

[2] A. ARNOLD AND M. NIVAT, *Controlling behaviours of systems: Some basic concepts and some applications*, in Mathematical Foundations of Computer Science 1980 (Lecture Notes in Computer Science, No. 88), 1980, Springer-Verlag, New York, pp. 113–122.

[3] J. R. BÜCHI, *On a decision method in restricted second order arithmetic*, in Logic, Methodology and Philosophy of Science, Proc. 1960 Internat. Congress, Stanford, CA, 1962, Stanford University Press, pp. 1–11.

[4] J. R. BÜCHI AND L. H. LANDWEBER, *Solving sequential conditions by finite-state strategies*, Trans. Amer. Math. Soc., 138 (1969), pp. 295–311.

[5] Y. CHOUEKA, *Theories of automata on ω-tapes: A simplified approach*, J. Comput. System Sci., 8 (1974), pp. 117–141.

[6] A. CHURCH, *Application of logic to the problem of circuit synthesis*, in Summer Institute for Symbolic Logic, Cornell University, Ithaca, NY, 1957, pp. 3–50.

[7] ———, *Logic, arithmetic and automata*, in Proc. Internat. Congress of Mathematicians, August 15–22, 1962, Djursholm, Sweden, 1963, Institut Mittag-Leffler, pp. 23–35.

[8] J. DE BAKKER, *The fixed point approach in semantics: Theory and applications*, in Foundations of Computer Science, J. de Bakker, ed., Mathematical Centre Tracts, Amsterdam, 1975, pp. 3–53.

[9] S. EILENBERG, *Automata, Languages and Machines*, Vol. A, Academic Press, New York, 1974.

[10] E. A. EMERSON, *Characterizing correctness properties of parallel programs using fixpoints*, in Internat. Colloquium on Automata, Languages and Programming, 1980 (Lecture Notes in Computer Science, No. 85), 1980, Springer-Verlag, New York, pp. 169–181.

[11] ———, *Automata, tableaux and temporal logics*, in Logics of Programs (Lecture Notes in Computer Science, Vol. 193), R. Parikh, ed., Springer-Verlag, Berlin, New York, June 1985, pp. 79–87.

[12] ———, *Temporal and modal logic*, in Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics, J. van Leeuwen, ed., Elsevier, The MIT Press, Cambridge, MA, 1990, pp. 995–1072.

[13] E. A. EMERSON AND C. S. JUTLA, *The complexity of tree automata and logics of programs (extended abstract)*, in 29th Ann. Sympos. on Foundations of Computer Science, White Plains, NY, Oct. 24–26, 1988, pp. 328–337.

[14] ———, *On simultaneously determinizing and complementing ω-automata (extended abstract)*, in IEEE Sympos. on Logic in Computer Science, Asilomar, CA, 1989, pp. 333–342.

[15] E. A. EMERSON AND C.-L. LEI, *Modalities for model checking: Branching time strikes back*, in Proc. 12th ACM Sympos. on Principles of Programming Languages, New Orleans, LA, 1985, pp. 84–96.

[16] ———, *Efficient model checking in fragments of the propositional mu-calculus (extended abstract)*, in Proc. of Sympos. on Logic in Computer Science, IEEE, Cambridge, MA, June 16–18, 1986, pp. 267–278.

[17] A. FUSAOKA, H. SEKI, AND K. TAKAHASHI, *A description and reasoning of plant controllers in temporal logic*, in Proc. 8th Internat. Joint Conference on Artificial Intelligence, Karlsruhe, Germany, Aug. 1983, pp. 405–408.

[18] C. GOLASZEWSKI AND P. RAMADGE, *Mutual exclusion problems for discrete event systems with shared events*, in Proc. 27th IEEE Conference on Decision and Control, Austin, TX, Dec. 7–9, 1988, pp. 234–239.

[19] Y. GUREVICH AND L. HARRINGTON, *Trees, automata and games*, in Proc. of Sympos. on the Theory of Computing, ACM, San Francisco, CA, May 5–7, 1982, pp. 60–65.

[20] R. HOSSLEY AND C. RACKOFF, *The emptiness problem for automata on infinite trees*, in Proc. of Switching and Automata Theory Sympos., IEEE, University of Maryland, Oct. 1972, pp. 121–124.

[21] D. KOZEN, *Results on the propositional μ-calculus*, Theoretical Comput. Sci., 27 (1983), pp. 333–354.

[22] R. KURSHAN, *Testing Containment of ω-Regular Languages*, Tech. Rep., AT&T Bell Laboratories, Murray Hill, NJ, Oct. 1986.

[23] ———, *Reducibility in analysis of coordination*, in Discrete Event Systems: Models and Applications, IIASA Conference, Sopron, Hungary, Aug. 3–7, 1987, (Lecture Notes in Control and Information Sciences, Vol. 103), P. Varaiya and A. Kurzhanski, eds., 1988, Springer-Verlag, New York, pp. 19–39.

[24] L. H. LANDWEBER, *Synthesis algorithms for sequential machines*, in Information Processing 68, 1969, North-Holland, Amsterdam, pp. 300–304.

[25] J.-L. LASSEZ, V. NGUYEN, AND E. SONENBERG, *Fixed point theorems and semantics: A folk tale*, Inform. Process. Let., 14 (1982), pp. 112–116.

[26] R. MCNAUGHTON, *Testing and generating infinite sequences by a finite automaton*, Inform. and Control, 9 (1966), pp. 521–530.

[27] D. E. MULLER, *Infinite sequences and finite machines*, in Proc. 4th Annual Sympos. on Switching Circuit Theory and Logical Design, IEEE, Chicago, IL, Oct. 1963, pp. 3–16.

[28] A. NERODE, A. YAKHNIS, AND V. YAKHNIS, *Concurrent programs as strategies in games*, in Logic from Computer Science: Proceedings of a Workshop held November 13-17, 1989, Mathematical Sciences Research Institute Publications Vol. 21, 1992, Springer, Berlin, New York, pp. 405–479.

[29] A. PNUELI AND R. ROSNER, *On the synthesis of a reactive module*, in Proc. 16th Annual Sympos. on Principles of Programming Languages, Association for Computing Machinery, Austin, TX, Jan. 1989, pp. 179–190.

[30] V. PRATT, *A decidable mu-calculus: Preliminary report*, in Proc. 22nd Annual Sympos. on Foundations of Computer Science, IEEE, Nashville, TN, Oct. 28–30, 1981, pp. 421–427.

[31] M. O. RABIN, *Decidability of second-order theories and automata on infinite trees*, Trans. Amer. Math. Soc., 141 (1969), pp. 1–35.

[32] ———, *Weakly definable relations and special automata*, in Mathematical Logic and Foundations of Set Theory, Y. Bar-Hillel, ed., North-Holland, Amsterdam, 1970, pp. 1–23.

[33] ———, *Automata on Infinite Objects and Church's Problem*, Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics No. 13, American Mathematical Society, Providence, RI, 1972; Lectures from the CBMS Regional Conference held at Morehouse College, Atlanta, GA, September 8–12, 1969.

[34] P. RAMADGE AND W. WONHAM, *Supervisory control of a class of discrete event processes*, SIAM J. Control Optim., 25 (1987), pp. 206–230.

[35] ———, *The control of discrete event systems*, Proc. IEEE, 77 (1989), pp. 81–98.

[36] P. J. RAMADGE, *Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata*, IEEE Trans. Automatic Control, 34 (1989), pp. 10–19.

[37] S. SAFRA, *On the complexity of ω-automata*, in Proc. 29th Annual Sympos. on the Foundations of Computer Science, White Plains, NY, Oct. 24–26, 1988, pp. 319–327.

[38] R. STREETT, *A Propositional Dynamic Logic of Looping and Converse*, Tech. Rep. TR-263, MIT Laboratory for Computer Science, Cambridge, MA, 1981.

[39] A. TARSKI, *A lattice-theoretical fixpoint theorem and its applications*, Pacific J. Math., 5 (1955), pp. 285–309.

[40] J. THISTLE AND W. WONHAM, *On the synthesis of supervisors subject to ω-language specifications*, in Proc. of the 1988 Conference on Information Sciences and Systems, Princeton University, Princeton, NJ, March 1988, pp. 440–444.

[41] ———, *Control of ω-automata, Church's problem, and and the emptiness problem for tree ω-automata*, in Computer Science Logic: 5th Workshop, CSL '91, Berne, Switzerland, October 1991, Proceedings (Lecture Notes in Computer Science, Vol. 626), E. Börger, G. Jäger, H. K. Büning, and M. Richter, eds., Springer-Verlag, Berlin, Heidelberg, 1992, pp. 367–381.

[42] ———, *Supervision of infinite behaviour of discrete-event systems*, SIAM J. Control Optim., 32 (1994), pp. 1098–1113, this issue.

[43] J. G. THISTLE, *Control of Infinite Behaviour of Discrete-Event Systems*, Ph.D. thesis, University of Toronto, Toronto, Canada, Jan. 1991; Systems Control Group Report No. 9012, Systems Control Group, Dept. of Electrical Engineering, University of Toronto, January

1991.

[44] W. THOMAS, *Automata on infinite objects*, in Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics, J. van Leeuwen, ed., Elsevier, The MIT Press, Cambridge, MA, 1990, pp. 134–191.

[45] M. Y. VARDI, *A temporal fixpoint calculus (extended abstract)*, in Proc. 15th Annual ACM SIGACT-SIGPLAN Sympos. on Principles of Programming Languages, San Diego, CA, Jan. 1988, pp. 250–259.

[46] ———, *Verification of concurrent programs: The automata-theoretic framework*, Ann. Pure Appl. Logic, 51 (1991), pp. 79–98.

[47] M. Y. VARDI AND P. WOLPER, *An automata-theoretic approach to automatic program verification (preliminary report)*, in Proc. 1986 IEEE Sympos. on Logic in Computer Science, Cambridge, MA, June 16–18, 1986, pp. 332–344.

[48] P. WOLPER, *Temporal logic can be more expressive*, Inform. Control, 56 (1983), pp. 72–99.

[49] H. WONG-TOI AND D. L. DILL, *Synthesizing processes and schedulers from temporal specifications*, in Computer-Aided Verification (Proc. of the CAV 90 Workshop) DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 3, American Mathematical Society, Providence, RI, 1991, pp. 272–281.

[50] A. YAKHNIS AND V. YAKHNIS, *Extension of Gurevich-Harrington's restricted memory determinancy theorem*, Ann. Pure Appl. Logic, 48 (1990), pp. 277–297.