MARKOV DECISION PROCESSES AND REGULAR EVENTS
(Extended Abstract)

Costas Courcoubetis[*]
Mihalis Yannakakis

AT&T Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

Desirable properties of the infinite histories of a finite state Markov Decision Process are specified in terms of a finite number of events represented as ω-regular sets. An infinite history of the process produces a reward which depends on the properties it satisfies. We investigate the existence of optimal policies and provide algorithms for the construction of such policies.

1. INTRODUCTION

Markov Decision Processes (MDP's) have been widely used to model systems in which the evolution is controlled by available decisions which can be taken when the system is in certain states. Such processes with finite sets of states and decisions have been widely investigated, see [D70], [Ro83]. A finite state MDP consists of a Markov chain with transition probabilities controlled by the available decisions. The traditional cost structure associated with such models is to assume that at each time the process is in a particular state and a certain action is taken, a reward depending on that state and action is obtained. There is a large body of theory developed for investigating the optimal operation of such systems, where one seeks to maximize performance measures such as the expected reward per unit time or the total discounted reward for some given discount. The formulation of such cost structures makes dynamic programming a natural framework for solving the above optimization problems.

In this paper we are interested in designing optimal decision strategies for MDP's under a new type of cost structure. Consider a system that is modelled as a finite-state Markov Decision Process. We would like to control the operation of the system so that it satisfies certain properties. Our definition of "property" is quite general: we view the infinite history of the MDP as an infinite word over the state space of the MDP, and allow a property to be any ω-regular (or rational) set over this alphabet. In general, we may have a number of such properties that we would like to satisfy. We do not assume that the properties are mutually consistent, that is in general it may be impossible to satisfy all of them simultaneously, and one could assign different importance in the system having histories satisfying the different properties. Then, if one associates with each infinite history a reward which depends on the properties that the history satisfies, one would like to design a decision

[*] Supported in part by ESPRIT BRA project SPEC.

policy that maximizes the expected reward. This is the problem we investigate in this paper.

We are given a finite state MDP and a finite number of ω-regular properties. We assume that these properties are given in terms of the corresponding automata on infinite words, called Buchi automata. Every set of properties has an associated (nonnegative) reward, which is produced by a history if it satisfies the properties in the set. We assume that the reward structure satisfies the following monotonicity constraint: if $R \subset Q$ are two sets of properties, then the reward associated with Q is at least as large as the reward associated with R. This constraint corresponds to the fact that we would like to satisfy as much as possible the given properties. We seek a policy which maximizes the expected reward. The class of policies over which we optimize includes policies that randomize and are allowed to use complete information about the past history of the system. We devise an algorithm which computes the maximum reward that can be produced and constructs an optimal policy. The time complexity of our algorithm is polynomial in the size of the MDP and exponential in the sizes of the automata that represent the properties. The exponential dependence of the algorithm is unavoidable; even a very special case of our problem is hard for deterministic exponential time [CY88].

The optimal decision policy does not necessarily take the same action every time it passes through the same state, that is, the best possible action at every time depends not only on the current state of the process but also on the past history. However, we show that it suffices to remember only a bounded amount of information about the past history. From the given MDP M and the given set of properties, we construct a new equivalent MDP M' which is defined on a larger state space, is coupled with M (in the sense that it is defined on the same probability space), and such that M' has a memoryless optimal decision policy, i.e., the policy depends on the history of the process only through the current state of M'.

One of the applications of the optimization problem that we solve in this paper (which was in fact our original motivation) is in the analysis of concurrent probabilistic finite-state programs. Randomization is by now an important tool in the design of efficient algorithms and protocols. It has been realized that randomization may often lead to algorithms with better complexity than deterministic ones, or even allow for the solution of problems (especially in distributed computation) that cannot be solved deterministically. We will define formally the model for concurrent probabilistic (finite state) programs and communication protocols in the next section, and observe that it is closely related to MDP's. Such programs and protocols are typically modelled as finite state transition systems that include both nondeterministic and probabilistic transitions. The nondeterminism is due to the asynchronous execution of the processes, or because some transitions are controlled by the environment (users, scheduler of the CPU, etc.). The probabilistic transitions are due to randomization in the protocol. Even in the case of protocols that do not use randomization, one may want to regard certain transitions as being probabilistic in order to analyse the behaviour of the protocol under some probabilistic assumptions about the environment (for example, a message is corrupted with certain probability, a buffer can be allocated with certain probability, etc.).

The wide applicability of probabilistic algorithms has motivated the interest in the development of formal, automatic methods to analyse such algorithms and verify that they meet their specifications. A simple language for specifying temporal properties of computations is propositional linear time temporal logic (PTL) [Pn77, Pn81]. More powerful languages, such as the Extended Temporal Logics (ETL) of [Wo83, WVS83] have been also proposed. It is known that the sets of computations (infinite histories) that satisfy properties expressed in the above logics are ω-regular: from a given formula ϕ of PTL (or ETL) one can construct an automaton A (of size exponential in the size

of ϕ) which accepts the computations that satisfy ϕ [WVS83]. To verify whether a concurrent probabilistic program P meets a specification ϕ, one assumes a worst-case scenario for the nondeterministic transitions controlled by the environment. That is, one considers the environment as a malicious adversary who tries to cause the program to violate its specification.

A number of papers in recent years have investigated the verification problem for such programs, i.e., testing whether the program meets a desired specification with probability 1, have introduced methods for solving this problem, proving tight upper and lower bounds on its complexity [LS82, HS84, V85, PZ86, VW86, CY88]. However, many times, probabilistic programs are supposed to satisfy certain requirements only with some given probability p which may be smaller than one. The existing techniques deal only with the *qualitative* aspects of probabilistic verification, and thus are not capable of verifying such programs. The problem of verifying such programs can be posed naturally as an optimization problem for MDP's: given a MDP P (the program) and a specification ϕ, find the optimal decision policy (worst possible nondeterminism) that maximizes the probability that the infinity history satisfies $\neg\phi$ (i.e., the process misbehaves). We can solve this problem by first constructing an automaton A for the formula $\neg\phi$ and then applying our algorithm for maximising the probability of a (single) regular event. The complexity of this algorithm is doubly exponential in the size of ϕ and polynomial in the size of the program. One cannot expect to do much better since a double exponential lower bound holds even for the qualitative verification problem [CY88] (determining whether the maximum probability of $\neg\phi$ is 0). Similarly, our methods can be used to perform a quantitative analysis on the behaviour of a protocol under probabilistic assumptions on certain parts of the environment.

The present abstract is organized as follows. In Section 2 we review briefly the necessary background on ω-regular sets, automata over infinite words, and MDP's. We define a model for an MDP which has the actions as explicit transitions; we call this model a *controlled Markov chain.* In Section 3 we deal with the case of a single regular event. We prove some structural properties of automata on infinite words and controlled Markov chains and then use these properties to solve the optimization problem. In Section 4 we deal with the case of many events. First, we solve the special case when the events are disjoint, and then finally address the general case, and analyse the complexity of our algorithm. In Section 5 we provide some extensions of the results and point out interesting directions for further research.

2. BACKGROUND

2.1. Automata on infinite words and ω-regular sets

A transition table is a tuple $\tau = (\Sigma, S, \rho)$, where Σ is the finite alphabet, S is a finite set of states, and $\rho : S\times\Sigma \to 2^S$ is the transition function. A state is *deterministic* if $|\rho(s,a)|\leq 1$ for all letters $a\in\Sigma$; the table τ is deterministic if all the states are. A run of τ over a finite word $w = a_1 \cdots a_n$ over Σ is a sequence of states $\mathbf{s} = s_0, \ldots, s_n$ such that $s_{i+1}\in\rho(s_i,a_i)$ for $0\leq i\leq n-1$. A run of τ over an infinite word $w = a_1 a_2 \cdots$ is an infinite sequence of states $\mathbf{s} = s_0, s_1, \cdots$ such that $s_{i+1}\in\rho(s_i,a_i)$ for $i\geq 0$. For an infinite run $\mathbf{s}$, the set $inf(\mathbf{s})$ is the set of states that repeat infinitely often in $\mathbf{s}$, i.e., $inf(\mathbf{s}) = \{v \mid |\{i: s_i = v\}| = \infty\}$. For the table τ and a state s, we denote by $det(\tau,s)$ the deterministic table that results from applying the usual subset construction to τ starting from state s. If we do not specify an initial state s, $det(\tau)$ denotes the deterministic table defined over all states in 2^S corresponding to τ.

Let Σ^ω be the set of infinite words over the alphabet Σ. An ω-automaton A consists of a table

$\tau_A = (\Sigma, S, \rho)$, a set of starting sates $S_0 \subseteq S$, and an acceptance condition. The automaton is deterministic if τ is deterministic and $|S_0| = 1$. In the usual automata on finite strings, acceptance of a run is determined by the final state of the run; in the case of ω-automata and infinite runs $\mathbf{s}$, acceptance is determined by its infinity set $inf(\mathbf{s})$. There are several ways in which one can specify which infinity sets are accepting and which ones are not. The simplest one is the Buchi acceptance condition. A Buchi automaton $A = (\tau_A, S_0, F)$, $F \subseteq S$, is an ω-automaton with the following acceptance condition. A accepts an infinite word w if there is a run $\mathbf{s}$ of τ_A over w such that $\mathbf{s}$ starts with a state in S_0 and some state in F repeats in $\mathbf{s}$ infinitely often, that is, $inf(\mathbf{s}) \cap F \neq \emptyset$.

We define as $A \subseteq \Sigma^\omega$ the set of infinite words accepted by the ω-automaton A. Nondeterministic Buchi automata accept exactly the ω-regular languages; deterministic Buchi automata accept a proper subset. Since in the rest of the paper we will be using only automata with the Buchi type of acceptance condition, we will refer to them as simply automata. We will usually denote an automaton and the language that it accepts by the same symbol.

2.2. Markov Decision Processes and controlled Markov chains

For a Markov Decision Process we use the standard definition found in [D70]. Let Y be the finite set of states of the system. Associated with every state $i \in Y$ there is a (finite) set K_i of possible actions. A decision policy is a prescription for choosing actions at each point in time. Let $\{Y_j,\ j=0,1,\cdots\}$ be the sequence of states of the process, let $\{A_j,\ j=0,1,\cdots\}$ be the sequence of actions taken, and let the history of the system up to time t be $H_t = Y_0, A_0, \ldots, Y_t, A_t$. Then a policy u is a set of functions $\{p_a(H_{t-1}, Y_t),\ a \in K_{Y_t},\ t=0,1,\cdots\}$ satisfying $0 \leq p_a \leq 1$ and $\sum_a p_a = 1$. That is, a policy chooses the probability of the next action based on the past history of the system.

If the system is in state i and action a is chosen, then the system transitions according to a given set of transition probabilities $\{q_{ij}(a) : j \in Y\}$. Given the initial distribution $P(Y_0 = i)$ and a policy u, the stochastic process $\{(Y_t, A_t),\ t=0,1,...\}$ is called a *Markov Decision Process* (MDP).

Let $K = \cup_i K_i$ be the set of all actions and consider any mapping $V : Y \cup K \rightarrow \Sigma$, where Σ is some finite alphabet. Let $\mathbf{H}$ denote an infinite history of the system and denote by $V(\mathbf{H})$ the sequence $V(Y_0), V(A_0), \cdots$. We call the above sequence the *valuation* of the sequence $\mathbf{H}$. Desired properties of the MDP are specified in terms of ω-regular languages A_i, $i=1, \ldots, m$, over the alphabet Σ. If the valuation $V(\mathbf{H})$ of a history $\mathbf{H}$ is in the language A_i, we will say that the history $\mathbf{H}$ satisfies property A_i. The cost structure we use is defined as follows. Let R denote any subset of the set of indices $\{1, \ldots, m\}$, and define the set

$$A_R = \cap_{i \in R} A_i \cap_{i \notin R} A_i^c, \tag{2.1}$$

where A^c is the complement of A. Clearly the sets A_R corresponding to different R's are disjoint. For every subset R of indices there is an associated reward k_R. A history $\mathbf{H}$ produces reward k_R iff its valuation is in A_R, i.e., the set of (indices of) properties it satisfies is exactly R. We assume that the k_R's are non negative and satisfy the following monotonicity property

$$R \subseteq Q \quad \text{implies} \quad k_R \leq k_Q. \tag{2.2}$$

In measuring the complexity of our algorithm, we do not assume that the rewards k_R are listed as part of the input for all possible 2^m subsets of indices, but allow that they be implicitely specified. For example, a special case is when there is a positive weight w_i associated with every property A_i,

and the reward produced by a history is the sum of the weights of the properties that it satisfies. Our problem is to design an optimal policy u which maximizes the expected reward $\sum_R k_R P_u(V(\mathbf{H}) \in A_R)$, where P_u is the probability measure defined by the MDP with policy u over the space Σ^ω.

In this paper we will use an equivalent formulation for an MDP. In this formulation, actions are explicitly modeled as special transitions. This provides for a uniform treatment of actions and state transitions in the above problem. We use the term controlled Markov chain to refer to this formulation, which is very similar to the notion of concurrent Markov chain introduced in [V85] to model finite state concurrent probabilistic programs. Informally, a controlled Markov chain has two types of states, the randomizing states and the control states. When the chain is at a randomizing state, a transition is chosen randomly according to a probability transition matrix; when the chain is at a control state, a transition is chosen by a *policy*, possibly depending on the history of the chain. Formally, a *controlled Markov chain M* consists of a finite set X of states, which is partitioned into a set N of *control* states and a set R of *randomizing* states; a set $E \subseteq X \times X$ of arcs (transitions); an assignment of nonzero probabilities to arcs coming out of randomizing states, so that the probabilities on the arcs coming out of each randomizing state sum to 1; an initial probability distribution, again summing to 1. Let $\{X_t,\ t=0,1,\cdots\}$ be the sequence of observed states, and let $H_t = X_0, \ldots, X_t$ be the history of the system up to time t. A *policy* is a set of functions $\{p_{ij}(H_t),\ i=X_t \in N,\ (i,j) \in E,\ t=0,1,\cdots\}$, satisfying $0 \le p_{ij} \le 1$, and $\sum_j p_{ij} = 1$. The interpretation of this definition is that at each time t, if the system is in a control state, the policy chooses based on the past history the probabilities for the transitions out of the present state. For each policy u, the process X_t, $t=0,1,\cdots$, is not Markov since at time t the distribution of X_t depends on the entire past history of the process. (We can view the process as a Markov chain on an infinite state space, namely X^*.)

Clearly, a controlled Markov chain corresponds to an MDP. Conversely, from an MDP we can construct an equivalent controlled Markov chain as follows. For each state $i \in Y$ of the MDP with associated set of actions $K_i = \{a_{i1}, \ldots, a_{ik_i}\}$, we have a control state $i \in N$, randomizing states $(i,a_{i1}), \ldots, (i,a_{ik_i}) \in R$, and the control transitions $i \to (i,a_{i1}), \ldots, i \to (i,a_{ik_i})$. From every randomizing state (i,a_{ik}) we have randomizing transitions to all control states j with transition probabilities $p((i,a_{ik}) \to j) = q_{ij}(a_{ik})$. A valuation mapping V from the states and actions of the MDP to an alphabet Σ can be translated in the straightforward way to a valuation from the states of the controlled Markov chain to Σ.

Let A be an automaton over the alphabet Σ, and let M be a controlled Markov chain with state space X and with valuation mapping $V: X \to \Sigma$. We can define another automaton A' over the alphabet X such that any infinite trajectory $\mathbf{X}$ of M is accepted by A' if and only if its valuation $V(\mathbf{X})$ is accepted by A; the automaton A' has the same states as A and for each state s and letter $x \in X$ its transition function is defined to be $\rho'_A(s,x) = \rho_A(s, V(x))$ where ρ_A is the transition function of A. Thus, for any policy u, $P_u(A) = P_u(A')$. A consequence of this observation is that we can assume without loss of generality that the alphabet Σ coincides with the state space X of the controlled Markov chain. We can now state our optimization problem as follows.

Problem: Given a controlled Markov chain over the state space X, the automata A_i, $i=1, \ldots, m$, over the alphabet X, and the non negative constants k_R, $R \subseteq \{1, \ldots, m\}$, satisfying (2.2), find a policy u which maximizes the reward function $\sum_R k_R P_u(\mathbf{X} \in A_R)$.

This is the problem which we investigate in the subsequent sections. In the rest of the paper we refer to controlled Markov chains as simply Markov chains.

2.3. Some examples

We give some examples of ω-regular sets and the corresponding automata. Let X be the alphabet, x_1, $x_2 \in X$, and as before $\mathbf{X}$ denotes an infinite word over X.

A_1 = {$\mathbf{X} | x_1$ appears infinitely often },

A_2 = {$\mathbf{X}$| at all times, if x_1 appears, then x_2 eventually appears at some future time },

A_3 = {$\mathbf{X}$| every four steps, x_1 appears at least once }.

The next set A_4 describes the erroneous behavior of a communication protocol and is borrowed from [ACW86]. This behavior corresponds to the sequence of messages which the receiver part of the protocol gives as an output to the end user. There is an arbitrary number of messages two of which are distinct, the rest of the messages being undistinguishable. At every step, the receiver outputs a message or performs a "pause" operation. The above system is described with the alphabet {r_1, r_2, r, pause}, where r_1 and r_2 correspond to the two distinguished messages, whereas all other messages are mapped to the letter r. The pause operation corresponds to the letter "pause". The set of correct histories for this system is the set of all histories for which r_1 and r_2 occur exactly once, and r_1 occurs before r_2. The set A_4 is the complement of the above set. If we model the protocol as an MDP, then, maximizing the probability of A_4 in this context will provide the upper bound for the probability of incorrect behaviour of the protocol under all possible decisions (by the users, the cpu scheduler, and all other nondeterministic factors like buffer overflows at the worst possible moment, etc.).

The automata for the above sets are shown in Figure 1.

3. MAXIMIZING THE PROBABILITY OF A REGULAR EVENT

In this section we deal with a controlled Markov chain M having state space X and a single automaton A defined over the alphabet X. We want to construct a policy that maximises the probability that the history is accepted by A. We shall construct a new controlled Markov chain M' (defined on the same probability space as M), and we shall identify a distinguished set T of *target* states of M' such that the maximum probability that a history of M is accepted by the automaton A is equal to the maximum probability that a history of M' hits a state from T; the latter probability can be computed by Linear Programming techniques. The optimal policy works in two phases. In the first phase it tries to make M' hit a target state; if this phase succeeds, then it switches to another strategy which ensures with probability one that the history produced is accepted by A.

The technical development involves a detailed analysis of the structure of ω-automata and their interaction with controlled Markov chains. The details are rather involved; we will outline below the basic ideas.

We can assume without loss of generality that the automaton A has a unique initial state, say s_0, and that it is completely specified, i.e., has a transition from every state on each letter of the alphabet X. We view M also as an automaton over the same alphabet X which generates infinite words. A transition of M from state v to a state x is labeled with the letter x (the tail of the transition); that is, the transition function ρ_M of M from state v on letter $x \in X$ is: $\rho_M(v,x) = \{x\}$ if M

has a transition from v to x, and is $\emptyset$ otherwise. Note that under this definition, M is a deterministic automaton, but it may not be completely specified. The infinite word (over the alphabet X) generated by a trajectory of M is identical to the trajectory, except that the first state of the trajectory is missing. For this reason, we first add a new dummy initial state x_0, and add a transition from x_0 to every state v with the probability of the transition equal to the probability of v in the (old) initial distribution (of course, if this probability is 0, then the transition $x_0 \rightarrow v$ is not included.) We want to maximize the probability that the new chain M generates a word accepted by A.

We can take now the product of M and A as two automata over the same alphabet X. If S is the set of states of A, then the set of states of the product transition table $\tau_{M\times A}$ is $X\times S$. On letter $x\in X$, a state (v,s) of the product goes to all states (x,s') such that $s'\in \rho_A(s,x)$ if M has a transition from v to x, and goes nowhere otherwise. Note that all transitions of $\tau_{M\times A}$ coming into a state with first component x are on letter x. Every (finite or infinite) run of $\tau_{M\times A}$ corresponds to a unique path in M; just project every state of the run on its first component. Conversely, every path of M corresponds to at least one run of $\tau_{M\times A}$. We consider $\tau_{M\times A}$ as an automaton with initial state (x_0,s_0) and with set of accepting states $X\times F$, where F is the set of accepting states of A. Observe that a trajectory $\mathbf{X} = x_0x_1x_2\cdots$ of M (starting from the new initial state x_0) corresponds to some accepting run of $\tau_{M\times A}$ starting from state (x_0,s_0) if and only if the infinite word $x_1x_2\cdots$ that it generates is accepted by the automaton A. The automaton $\tau_{M\times A}$ accepts the intersection of the languages of M and A.

Consider the deterministic table $\det(\tau_{M\times A},(x_0,s_0))$ obtained from $\tau_{M\times A}$ by applying the usual subset construction starting from state (x_0,s_0). It is easy to see that every state of this deterministic table is a set of pairs (x,s) that have the same first component, i.e., it is of the form $\{x\}\times Q$ for some state $x\in X$ of the Markov chain and some set Q of states of the automaton A. We define the Markov chain M' on this transition table; that is, the states and transitions of M' are those of $\det(\tau_{M\times A},(x_0,s_0))$ and the initial state of M' is (x_0,s_0). A state $\{x\}\times Q$ of M' is a control state or a randomising state depending on whether x is a control or randomizing state of M. If x is a randomising state and has a transition to a state x' in M, then the state $\{x\}\times Q$ of M' has a corresponding transition to a unique state $\{x'\}\times Q'$ with first component x'; we assign to this transition of M' the same probability as the transition $x \rightarrow x'$ of M. We think of M' as being constructed on the same probability space as M, so that for each infinite trajectory $\mathbf{X}$ of M there is a unique corresponding infinite trajectory $\mathbf{Y}$ of M'; $\mathbf{Y}$ is simply the trajectory of M' whose projection on the first component is equal to $\mathbf{X}$. A point in our probability space corresponds to a pair $(\mathbf{X}, \mathbf{Y})$ of coupled trajectories of M and M'

To define the set T of target states of M', we need first the following definition.

Definition 3.1: A state (x,s), $x\in X$, $s\in S$ is *controllably recurrent* if in $det(\tau_{M\times A},(x,s))$ there exists a strongly connected subset C with a state $y\in C$, $(x,s)\in y$, such that
(a) There is no transition out of C corresponding to a randomizing transition in M, and
(b) There is a finite string $\gamma\in X^*$ such that the run of $det(\tau_{M\times A},(x,s))$ over γ starting at y visits only states in C, and the run of $det(\tau_{M\times A},(x,s))$ over γ starting at $\{(x,s)\}$ ends inside C.

The following proposition summarizes the important properties of controllably recurrent states.

Proposition 3.1: Let (x,s) be any state of $\tau_{M\times A}$, and let Ψ denote the event that a trajectory of M corresponds to a run of $\tau_{M\times A}$ that repeats (x,s) infinitely often. There is a policy u such that $P_u(\Psi) > 0$ if and only if (x,s) is controllably recurrent and is reachable from the initial state (x_0,s_0) of $\tau_{M\times A}$. □

A corollary of this proposition is that the maximum probability $\max_u P_u(A)$ that M generates a trajectory in (the language of) A is positive if and only if the initial state (x_0,s_0) of $\tau_{M\times A}$ can reach an accepting controllably recurrent state (x,f) (i.e., with $f\in F$). We define now the set T of target states of M'.

Definition 3.2: If (x,f) is an accepting controllably recurrent state of $\tau_{M\times A}$, then let $V(x,f)$ be the set of those states of $\det(\tau_{M\times A},(x,f))$ which belong to some strongly connected subset C such that the conditions of Definition 3.1 are satisfied for (x,f) and C. Let V be the union of the sets $V(x,f)$ for all accepting controllably recurrent states (x,f). A state y of M' is a *target state* iff there is a member z of V such that $z \subseteq y$.

The following two lemmas give the inportant properties of the set T of target states.

Lemma 3.1: Let $\gamma=x(0)\cdots x(n)$, $x(0)=x_0$, be any finite sequence of states of M for which the run of $det(\tau_{M\times A},(x_0,s_0))$ starting from $\{x_0,s_0\}$ ends in some state in T. Then, if the Markov chain M performs as its first n transitions the transitions corresponding to γ, there exists a strategy μ which is used from time n onwards and produces with probability 1 an infinite trajectory accepted by the automaton A (this trajectory has γ as its prefix). □

For the purposes of this section, we need only the converse of Lemma 3.1; i.e., under any policy u, the trajectories of M that are accepted by A almost surely correspond to trajectories of M' that hit T. The following stronger converse is true, and will be needed in the general case of many events.

Lemma 3.2: Let $\mathbf{X}$ be a trajectory of M starting from x_0 and let $\mathbf{Y}$ be the corresponding trajectory of M' starting from $\{(x_0,s_0)\}$. Then, for any policy u, the probability that $\mathbf{X}\in A$ and that $\mathbf{Y}$ does not hit some state in T infinitely often is equal to zero. □

For any policy u, let $\zeta_u(T)$ be the first time that the trajectory $\mathbf{Y}$ of M' hits T (recall that M' is defined on the same probability space as M). From Lemma 3.2 it follows that $P_u(A)\le P_u(\zeta_u(T)<\infty)$. For any policy π and its corresponding stopping time $\zeta_\pi(T)$, define $u(\pi)$ to be the policy which immitates π until the stopping time $\zeta_\pi(T)$ (forever if $\zeta_\pi(T)=\infty$) and then uses strategy μ from that time on. It follows from Lemma 3.1 that for any policy π we have $P_{u(\pi)}(A)=P_\pi(\zeta(T)<\infty)$. Combining this with the previous inequality, we have the following optimality theorem.

Theorem 3.1: If π^* is a policy that maximizes $P_\pi(\zeta_\pi(T)<\infty)$ over all policies π, then the corresponding policy $u(\pi^*)$ maximizes $P_u(A)$. □

The maximum probability $p^* = \max_u P_u(A)$ and the actions of the optimal policy during the first phase can be computed from the solution of a Linear Program constructed as follows. Let Y be the set of states of the chain M', and let $Y_N\subseteq Y$ denote the subset of the control states, $E(i)$ be the set of states to which there is a transition from state i, and $V(i)$ be the maximum probability over all policies that the chain starting from i will eventually hit T. We define the linear program LP1: minimize $\sum_{i\in Y} v(i)$ subject to the constraints (1) $v(i) \ge \sum_{j\in E(i)} p_{ij}v(j)$ for $i\in Y-Y_N$, $i\notin T$; (2) $v(i) \ge v(j)$ for $j\in E(i)$, $i\in Y_N$, $i\notin T$; (3) $v(i) \ge 0$ for $i\in Y$; (4) $v(i) = 1$ for $i\in T$.

Proposition 3.2: V is the optimal solution of LP1. In particular, $\max_u P_u(A)$ is given by the value of $v(x_0,s_0)$ in the optimal solution. □

4. THE OPTIMIZATION PROBLEM FOR MANY EVENTS

We will first analyse the case that the events (the desired properties) are disjoint. This case contains most of the probability-theoretic subtleties of the problem. Then we will proceed to the general case which requires further structural analysis of automata. The optimal policies of this section operate also on a new controlled Markov chain with a larger state space, and work in two phases. In the first phase the policy solves an optimal stopping time problem; when this phase finishes, the policy has decided on a subset of properties that it can probably achieve, and has given up on the rest of them. In the second phase, the policy switches to a strategy that ensures that the properties in the subset are satisfied with probability 1. It is critical that we can stop the first phase after finite expected duration, even though we deal with properties that depend on the entire infinite history, i.e., observing the system for any finite time may not be enough to tell whether it will satisfy a property or not.

4.1. Disjoint events

We are given a controlled Markov chain M', automata A_i, $i=1, \ldots, m$, corresponding to m disjoint regular sets and the positive scalars $k_1 \leq \cdots \leq k_m$. We want to find a policy which maximizes the reward $R_u = \sum_i k_i P_u(A_i)$.

For each automaton A_i let S_i be its state space and s_0^i its initial state. Let L be the transition table corresponding to the product of the tables $det(\tau_M)$, $det(\tau_{A_1}), \ldots, det(\tau_{A_m})$, with initial state the state $(x, s_0^1, \ldots, s_0^m)$. Clearly L is also deterministic and its states can be written in the form $z=(x, w_1, \ldots, w_m)$, $w_i \in 2^{S_i}$, with the interpretation that if the run of L over some word a starting from $(x_0, \{s_0^1\}, \ldots, \{s_0^m\})$ ends on z, then for each i, the run of $det(\tau_{M \times A_i}(x_0, s_0^i))$ over a will end on (x, w_i). We define the projection on the i'th component of the state z of L to be the state (x, w_i) of the corresponding table $det(\tau_{M \times A_i}(x_0, s_0^i))$. We let M' be the controlled Markov chain defined on the transition table L by associating the transition probabilities of M as we did in the case of one automaton in the previous section. We consider M and M' as being defined on the same probability space. When we refer to the reward R_u of a policy in M' or deal with the probability of events $P_u(A_i)$, since these events are in the probability space of the original chain M, we implicitely project the trajectories of M' on the component of the state corresponding to M.

Let Y be the state space of M'. We construct the sets $T_i \subseteq Y$, $i=1, \ldots, m$, such that a state z of M' is in T_i if the projection of z on its i'th component is one of the target states of the table $det(\tau_{M \times A_i}(x_0, s_0^i))$ corresponding to M and A_i as defined in the previous section. Although the events are disjoint, it may be the case that some states have their projections on two or more components belonging to the target sets of the corresponding tables; we make the T_i's disjoint by associating any state in Y which belongs to more than one T_i's by the previous construction, to the T_i with the largest index (i.e., the one with the largest reward).

We define the following two systems O_1, O_2. The first system O_1 is the original system consisting of the chain M' and the cost structure $R_u^1 = \sum_i k_i P_u(A_i)$. A policy in this system will be denoted by u and Y_t denotes the state of the chain at time t.

The second system O_2 consists of a different controlled Markov chain M'' and a new cost structure defined as follows. The new chain is identical to M' with the addition of the m absorbing states $y_1^*, \ldots, y_m^*$, and at each state in T_i we add a new decision for transitioning to y_i^*. In this system we associate k_i units of reward with the transitions $y \to y_i^*$, $y \neq y_i^*$, $i=1, \ldots, m$, and zero reward

with all remaining transitions. This implies that on every trajectory, reward will be produced only once, and this will occur when the system will for the first time transition to some state in $y_1^*, \ldots, y_m^*$. A policy in this system is denoted by π and Z_t denotes the state of the chain at time t. The reward structure in this system is the total expected reward obtained during the infinite execution of the system.

As we will show, an optimal policy in O_2 gives rise to an optimal policy in O_1. For each policy π of O_2, let ζ_π be the stopping time (in O_2) corresponding to the first time Z_t visits any state in the set $\{y_1^*, \ldots, y_m^*\}$. Define the policy $u(\pi)$ in O_1 as follows. For $t < \zeta_\pi$, the policy $u(\pi)$ immitates π; if π visits state y_i^*, then for $t \geq \zeta_\pi$, $u(\pi)$ uses strategy μ_i, as defined before. We shall show that if $\pi*$ is an optimal policy in O_2, then $u(\pi*)$ is an optimal policy for O1.

Lemma 4.1: For all policies π of O_2 we have $R^1_{u(\pi)} \geq R^2_\pi$. □

To prove a converse, we will argue that given any policy u in O_1 we can construct a policy π in O_2 which produces expected reward arbitrarily close to the reward produced by u. First, we need to introduce some new quantities. Let $\mathbf{Y}_s$ denote the history $Y_0, \ldots, Y_s$, and let $T_{>i} = \bigcup_{j>i} T_j$ ($T_{>m} = \emptyset$). For every policy u we define the function $g_u(\mathbf{Y}_s) = \sum_i k_i P_u$(after time s, $\mathbf{Y}$ will hit eventually some state in T_i and never visit any state in $T_{>i} \mid \mathbf{Y}_s$). Using the analysis of the previous section we can show that the reward is bounded from above by g_u applied to the empty history:

Lemma 4.2: Let s be an arbitrary time. Then the following holds. $\sum_i k_i P_u(A_i) \leq \sum_i k_i P_u(\mathbf{Y}$ hits i.o. T_i and does not hit i.o. $T_{>i}) \leq \sum_i k_i P_u($ after time s, $\mathbf{Y}$ hits eventually T_i and never again $T_{>i})$. The same inequalities also hold if we consider conditional probabilities on some arbitrary event. □

Given any positive scalar ε we construct the stopping time ψ_u of Y_t as follows. Every time s during which M' visits a state in $\bigcup_i T_i$ we compare $g_u(\mathbf{Y}_s)$ with $k_i + \varepsilon$, where i is such that $Y_s \in T_i$. If $g_u(\mathbf{Y}_s) < k_i + \varepsilon$ we define $\psi_u = s$ and stop. If no such s exists, then $\psi_u = \infty$. We call such a $\mathbf{Y}_\psi$ a *stopping history*. We define now the policy $\pi(u, \varepsilon)$ in O_2. This policy is identical to u up to time ψ_u. Then at time ψ_u it makes Z_t jump to the state y_i^* where i is such that $Y_{\psi_u} \in T_i$.

Lemma 4.3: For any policy u in O_1 and any positive scalar ε, we have $R^2_{\pi(u,\varepsilon)} \geq R^1_u - \varepsilon$. □

We can prove now the optimality theorem.

Theorem 4.1: If π^* is a policy which maximizes R^2_π in O_2, then the policy $u(\pi^*)$ maximizes R^1_u in O_1. □

4.2. The general case

We are given the sets A_i, $i = 1, \ldots, m$, in terms of their corresponding automata, and the non negative constants k_R, $R \subseteq \{1, \ldots, m\}$, satisfying (2.2). In principle we could solve the general case by reducing it to the disjoint case: for every subset R of indices we could have a property A_R as in equation (2.1) with associated reward k_R. Since the A_R's are obviously regular disjoint events, we could then use the method of the previous subsection. However, this approach would increase drastically the complexity: there are 2^m new properties, and for each one of them we would have to construct an accepting automaton; this would involve complementing some of the original automata and taking their product. In this subsection we will explore further the structure of automata to solve the general case with essentially the same complexity as the disjoint case.

Let L be the deterministic transition table introduced in the previous section, corresponding to the product $det(\tau_M)\times det(\tau_{A_1})\times\cdots\times det(\tau_{A_m})$, with initial state the state $(x,s_0^1,\ldots,s_0^m)$, and let M' be the corresponding Markov chain.

Definition 4.1: A strongly connected subset C of the table L *favors* the set A_R, $R\subseteq\{1,\ldots,m\}$, if the following conditions hold

(a) There is no randomizing transition out of C,

(b) for each $i\in R$, there exists a state $y^i=(x,y_1^i,\ldots,y_m^i)$, of C, an accepting state f^i of the automaton A_i, and a word $w^i\in X^*$, with the following properties

(b1) $f^i\in y_i^i$,

(b2) the run of L over w^i starting from y^i stays in C,

(b3) the run of L over w^i starting from $(x,y_1^i,\ldots,\{f^i\},\ldots,y_m^i)$ ends inside C. ($(x,y_1^i,\ldots,\{f^i\},\ldots,y_m^i)$ is the state which is identical to y^i exept on the i'th component, which is $\{f^i\}$).

We will say that a state $z=(x,z_1,\ldots,z_m)$ of M' *contains* another state $z=(x,z_1',\ldots,z_m')$ if $z_i'\subseteq z_i$, $i=1,\ldots,m$, and we denote this by $z'\subseteq z$. We define now the sets T_R of states of M' as follows. For each $R\subseteq\{1,\ldots,m\}$, T_R is the set of all states z of L such that z contains some state of a strongly connected subset favoring A_R. A state may belong to more than one sets T_R. The important fact is that the sets T_R have properties similar to those of the target sets T_i of Section 4.1. The following two lemmas generalize appropriately Lemmas 3.1 and 3.2.

Lemma 4.4: Let $\gamma=x(0)\cdots x(n)$, $x(0)=x_0$, be any finite sequence of states of M for which the run of L starting from $\{x_0,s_0^1,\ldots,s_0^m\}$ ends in some state in T_R. Then, if the Markov chain performs as its first n transitions the transitions corresponding to γ, there exists a strategy μ_R which is used from time n onwards and produces with probability one an infinite trajectory in the set $\cup\{A_Q \mid R\subseteq Q\}$ (this trajectory has γ as its prefix). □

Lemma 4.5: Let $\mathbf{X}$ be a trajectory of M starting from x_0 and let $\mathbf{Y}$ be the corresponding trajectory of M' starting from $\{(x_0,s_0^1,\ldots,s_0^m)\}$. Then, for any policy u, the probability that $\mathbf{X}\in A_R$ and that $\mathbf{Y}$ does not hit some state in T_R infinitely often is equal to zero. □

We can repeat now the same construction we did in the previous section and use the sets T_R, $R\subseteq\{1,\ldots,m\}$, instead of the sets T_i, $i=1,\ldots,m$. In this construction we associate with each set T_R an absorbing state y_R^*, and the transition to this state produces reward k_R. We define a system O_2 as in Section 4.1 and solve the optimal stopping time problem for this system. If π^* is the optimal policy for O_2, then the optimal policy $u(\pi^*)$ immitates π^* until it reaches some state y_R^*, and then switches to the corresponding strategy μ_R of Lemma 4.4. The properties of the T_i's used in the proofs of the previous section is that these sets satisfy Lemmas 3.1 and 3.2. By using Lemmas 4.4 and 4.5 instead of Lemmas 3.1 and 3.2 we can carry out a similar analysis and prove that the optimality theorem 4.1 holds in the general case as well. We can compute the maximum reward and the optimal policy that achieves it by solving a Linear Program. Let Y be the set of states of M', let $Y_N\subseteq Y$ be the subset of the control states, let $E(i)$ be the set of states to which there is a transition from state i, and let $V(i)$ be the maximum reward that can be achieved over all policies when the chain starts from state $i\in Y$. Let LP2 be the following Linear Program: minimize $\sum_{i\in Y}v(i)$ subject to the constraints (1) $v(i)\geq\sum_{j\in E(i)}p_{ij}v(j)$ for $i\in Y-Y_N$; (2) $v(i)\geq v(j)$ for $i\in Y_N$, $j\in E(i)$; (3) $v(i)\geq 0$ for $i\in Y$; (4) $v(i)\geq k_R$ for $i\in T_R$.

Proposition 4.1: V is the optimal solution of LP2. In particular, the optimum reward is given by the value in the optimal solution of the variable corresponding to the initial state of M'. □

4.3. Complexity

In the full paper we will see how to implement Definition 4.1 and find efficiently the target sets T_R. Once we have done this, we can solve the Linear Program LP2. In analysing the complexity, we include in the size of M the maximum length (in binary) of the transition probabilities of M and the rewards k_R. We measure the size of the given properties by the sum n of the states of the corresponding automata.

Theorem 4.2: The optimization problem can be solved in time polynomial in the size of the MDP and exponential (c^n for some constant c) in the size of the automata. □

5. EXTENSIONS AND OPEN PROBLEMS

We can extend the results in several directions. One extension is to impose some fairness restrictions and allow only policies that obey these restrictions. Suppose that a controlled Markov chain has a set N_F of its control states specified as the *fair* states. We define a policy of a controlled Markov chain as being *fair* if the following condition holds: for any infinite trajectory of the controlled Markov chain, if a fair state $w \in N_F$ appears infinitely often then all possible transitions out of w are taken infinitely often by the policy. We can use similar techniques to solve the optimization problem over policies that are restricted to be fair. Definitions 3.1 and 4.1 have to be modified in this case to reflect the fairness restrictions, but the rest of the development stays essentially the same. We postpone the technicals details for the full paper.

Another extension is is to allow for a more general definition of a valuation mapping (see Section 2). We could have that $V : X \rightarrow \Sigma \bigcup \{\varepsilon\}$, where ε is the empty string. This allows us to model invisible transitions by the system. One can easily see that the remark in the end of Section 2.2 still holds, i.e., we can construct an automaton A' for which we can use the identity mapping $\Sigma = X$. If we want to exclude histories which eventually become unobservable, we can modify slightly A' so that infinite runs which eventually perform only transitions labeled with ε are not accepted.

Some other interesting extensions do not seem to be as immediate. One problem is the corresponding minimization problem for regular events, or more generally, we may have a mix of properties some of which are desirable and some are undesirable. Of course, since regular sets are closed under complementation, we could use our techniques to solve this problem by complementing the undesirable properties. However, we do not know whether we can avoid an increase in complexity, i.e., whether we can solve this problem with the same efficiency as that of Theorem 4.2.

A more general problem is to develop a theory that combines sample path properties, such as the ones we consider here, with state dependent rewards as in the traditional cost structures of MDP's. For example, we would like to solve the classic MDP optimization problem conditioned on the event that the trajectories generated by the policies must satisfy certain sample path properties similar to the ones considered in this paper.

REFERENCES

[ACW86] S. Aggarwal, C. Courcoubetis and P. Wolper, "Adding Liveness Properties to Coupled Finite-State Machines", AT&T Bell Laboratories Technical Memorandum, to appear in *ACM TOPLAS.*

[BR85] F. Beutler and K. Ross, "Optimal Policies for Controlled Markov Chains with a constraint", *J. Math. Analysis and Appl.,* **112**, pp. 236-252, 1985.

[BR86] F. Beutler and K. Ross, "Time-Average Optimal Constrained Semi-Markov Decision Processes", *Adv. Applied Prob.,* **18**(2), pp. 341-359, 1986.

[CDFV88] R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya, "Supervisory Control of Discrete-Event Processes with Partial Information", *IEEE Trans. on Automatic Control,* **33**(3), pp. 249-260, March 1988.

[CY88] C. Courcoubetis and M. Yannakakis, "Verifying Temporal Properties of Finite-State Probabilistic Programs", *Proc. of 29th FOCS,* 1988, pp. 338-345, Oct. 1988.

[D70] C. Derman, *Finite-State Markovian Decision Processes,* Academic Press, New York, 1970.

[K83] L.C.M. Kallenberg, *Linear Programming and Finite Markovian Control Problems,* Mathematical Center Tracts, Amsterdam, 1983.

[McN66] R. McNaughton, "Testing and Generating Infinite Sequences by a Finite Automaton", *Information and Control,* **9**(1966), pp. 521-530.

[Pn81] A. Pnueli, "The Temporal Logic of Concurrent Programs", *Theoretical Computer Science* 13(1981), pp. 45-60.

[QS82] J. P. Queille, J. Sifakis, "Fairness and Related Properties in Transition Systems", Research Report #292, IMAG, Grenoble, 1982.

[Ra72] M. O. Rabin, "Automata on Infinite Objects and Church's Problem", *Proc. Regional AMS Conf. Series in Math.* 13(1972), pp. 1-22.

[Ro83] S. Ross, *Introduction to Stochastic Dynamic Programming,* Academic Press, New York, 1983.

[RV90] K. Ross and R. Varadarajan, "Markov Decision Processes with Sample Path Constraints; the Communicating Case", to appear in *Operations Research,* 1990.

[RW87] P. Ramadge and W.M. Wonham, "Supervisory Control of a Class of Discrete-Event Processes", *SIAM J. on Contr. and Optimization,* **25**(1), pp. 206-230, January 1987.

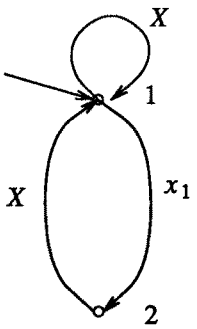
[S88] S. Safra, "On the Complexity of ω-automata", *Proc. of 29th FOCS,* 1988, pp. 319-327, Oct. 1988.

[V85] M. Vardi, "Automatic Verification of Probabilistic Concurrent Finite-State Programs", *Proc. of 26th STOC,* 1985.

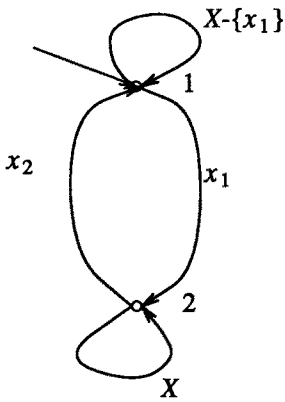
[VW86] M. Vardi and P. Wolper, "An automata-Theoretic Approach to Automatic Program Verification", *Proc. 1st Symp. on Logic in Computer Science,* 1986.

[WR87] W. M. Wonham and P. Ramadge, "On the Supremal Controllable Sublanguage of a Given Language", *SIAM J. on Contr. and Optimization,* **25**(3), pp. 637-659, May 1987.

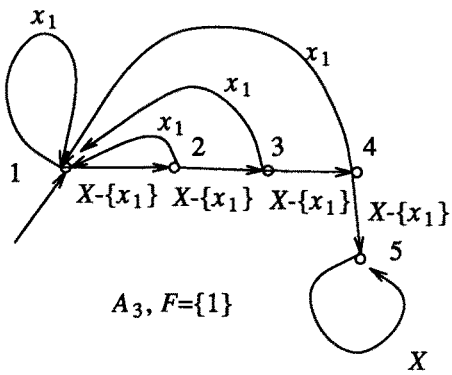
[WVS83] P. Wolper, M. Y. Vardi, A. P. Sistla, "Reasoning about Infinite Computation Paths", *Proc. of 24th IEEE Symp. on Foundations of Computer Science,* 1983, pp. 185-194.



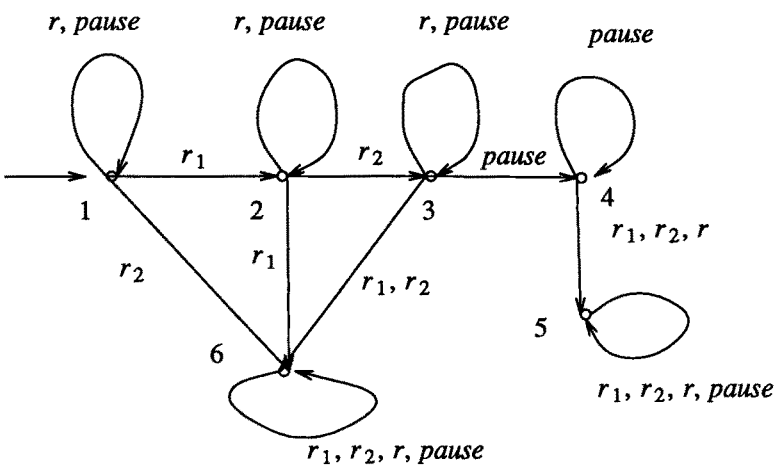
A_1, F={2}



A_2, F={1}



A_3, F={1}



A_4, F={1,2,4,6}

FIGURE 1