A formal characterization of maintenance goals

Mutsumi Nakamura and Chitta Baral

Dept. of Computer Science and Engg., Arizona State University, Tempe, AZ 85233
{mutsumi,chitta}@asu.edu

Marcus Bjäreland

Department of Computer and Info Sc., Linkoping University, S-581 83 Linkoping, Sweden
marbj@ida.liu.se

Abstract

The notion of maintenance often appears in the AI literature in the context of agent behavior and planning. In this paper we argue that earlier characterizations of the notion of maintenance is not intuitive to characterize the maintenance behavior of certain agents. We propose a different characterization of maintenance and distinguish it from earlier notions such as *stabilizability* and the temporal specification $\Box\Diamond f$. We present a simple algorithm to test maintainability and another algorithm that constructs a maintainble control. Noticing the incomparability of our notion of maintenance and the notion of stabilizability, we propose the notion of (k,l)-maintainability that generalizes both.

1 Introduction and Motivation

For agents in a static environment the goal of the agent is often to reach one of the states where certain conditions are satisfied. Such a goal is usually expressed by a propositional or first-order formula. Sometimes the goal requires constraining the path taken to reach one of the states. In that case the goal can be expressed by a temporal formula [BK98,NS00,BKT01]. Our concern in this paper is about agents in a dynamic environment. In that case the state of the world can change both due to the agent's actions and also due to the environment's actions. The agent's goal in a dynamic environment is often more

[1] A preliminary version of this work appears in AAAI'00. The current version differs substantially from the AAAI version and is almost written from scratch.

than just achieving a desired state, as after the agent has acted to reach a desired state the environment may change that state. In such a case a common goal of an agents is to 'maintain' rather than just 'achieve' certain conditions. The goal of maintaining certain conditions (or a set of states that satisfy these conditions) is referred to as *maintenance goals*. Although maintenance goals have been mentioned in the AI literature [WE94,KBSD97,BK98,Ort99] and have counterparts in stability theory of discrete event dynamic systems [OWA91,PB98,RW87b,RW87a,Son99], and active databases [CW90,NB00] we argue—in this paper—that their earlier characterizations are not adequate. We first propose a different characterization of maintainability and then propose a more general characterization that encompasses our initial characterization and also the past characterizations.

Suppose an agent's goal is to maintain a fluent f. A naive attempt to express it using temporal operators is the formula $\Box f$, where $\Box$ is the temporal operator *always* and $\Box f$ means that f is *true* in all the future states of the world. This is too strong a condition as maintaining inherently implies that things go out of shape and they have to be maintained back to shape. A better temporal logic representation of this goal is the formula $\Box\Diamond f$, where $\Diamond$ is the temporal operator *eventually*. Intuitively the formula $\Box\Diamond f$ is satisfied by an infinite trajectory of states of the form $s_0, s_1, s_2, \ldots$, if for any i, there exists a $j \geq i$ such that f is true in s_j. An agent's control is said to satisfy $\Box\Diamond f$ if all trajectories that characterize the evolution of the world due to the environment and the agent's control satisfy $\Box\Diamond f$. At first glance the formula $\Box\Diamond f$ seems to express the goal of maintaining f, as it encodes that if f becomes $false$ in any state in the trajectory then it becomes $true$ in a later state.

We consider $\Box\Diamond f$ to be also too strong a specification—in many situations—to express the intuitive notion of 'maintaining f'. Suppose f denotes the condition that the Inbox of a customer service department be empty. Here the environment makes f false by adding new requests to the Inbox while the agent tries to make f true by processing the messages in the Inbox and removing them from it. If the agent is diligent in processing the message in the Inbox and makes it empty every chance it gets we would then like to say that agent maintains the Inbox empty. But such a control does not satisfy the formula $\Box\Diamond f$ because there will be trajectories where the agent is overwhelmed by the environment and f never becomes *true*.

Another example in support of our intuition behind maintainability is the notion of maintaining the consistency of a database [CW90,NB00,WC96]. When direct updates are made to a database, maintaining the consistency of the database entails the triggering of additional updates that may bring about additional changes to the database so that in the final state (after the triggering is done) the database reaches a consistent state. This does not mean that the database will reach consistency if continuous updates are made to it and it is not given a chance to recover. In fact if continuous update requests are made then we may have something similar to denial service of attacks. In this case we can not fault the triggers saying that they do not maintain the consistency of the database. They do. It is just that they need to be given a window of opportunity or a respite from continuous harassment from the environment to bring about the additional changes necessary to make the database consistent. Same holds for maintaining a room clean. We can not fault the person who is coming to clean it if he or she is continually sent away because the room is being continuously used.

Another example is a mobile robot [Bro86,Mae91] which is asked to 'maintain' a state where there are no obstacles in its front. Here, if there is a belligerent adversary that keeps on putting an obstacle in front of the robot, then the robot can not get to a state with no obstacle in its front. But often we will be satisfied if the robot avoids obstacle in its front when it is not continually harassed. Of course, we would rather have the robot take a path that does not have such an adversary, but in the absence of such a path, it would be acceptable if it takes an available path and 'maintains' states where there are no obstacle in front.

The inadequacy of the expression $\Box\Diamond f$ in expressing our intuition about 'maintaining f' is because $\Box\Diamond f$ is defined on trajectories which do not distinguish between transitions due to agent's actions and environment's actions. Thus we can not distinguish the case (i) where the agent does its best to maintain f (and is sometimes thwarted by the environment) and can indeed make f true in some (say, k) steps if there is no interference from the environment during those steps; and (ii) the case where the agent really does not even try. We refer to (i) as k-maintainability in this paper. The expression $\Box\Diamond f$ can not express the idea of a window of opportunity (or window of non-interference) during which an agent can perform the actions necessary for maintaining. In fact none of the standard notion of temporal logics [CES86,MP92] which are defined on trajectories that do not distinguish between the cause behind the transitions (whether they are due to agent's actions or due to the environment) can express the idea behind k-maintainability.

In Section 4 we precisely define the notion of k-maintainability and distinguish it from the specification $\Box\Diamond f$ and the similar notion of stabilizability from discrete event dynamic systems. The rest of this paper is organized as follows. In Section 2 we present the background definitions of a system with an agent in an environment and define the notions of stability and stabilizability. In Section 3 we present a running example of a system with two buffers and use this example in illustrating the concepts of stabilizability and k-maintainability. In Section 5 we present algorithms to verify k-maintainability of a given control and to generate a maintainable control. In Section 6 we propose the notion of (k,l)-maintainability that generalizes both k-maintainability and stabilizability. Finally, in Section 7 we conclude, mention related work and some future directions.

2 Background: actions, states, control, and goals

In this paper we are concerned with goal-directed agents in a dynamic world. Such agents can perform actions that change the state of the world. Because of the dynamic nature of the world, certain actions beyond the control of the agents can happen and they also change the state of the world. The agent's job is to make the world evolve in a way consistent with a goal assigned to it. In this paper we assume that the agent follows a Markovian control policy to do its job. I.e., its control is a function from the set of states to the set of actions.

Let $\mathcal{A}$ be the set of actions which is the union of the set of agents actions $\mathcal{A}_{agent}$ and the set of environmental actions $\mathcal{A}_{env}$; $\mathcal{S}$ be the set of states; $\Phi : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ be a non-deterministic transition function that specifies how the state of the world changes in response to actions; and $poss : \mathcal{S} \rightarrow 2^{\mathcal{A}}$ be a function that describes which actions are possible in which states. (Normally Φ and $poss$ are specified in a representation language such as in [GL92,FN71,Rei01].) The possibility of an action has different meaning for depending on whether it is an agent's action or whether it is an environmental action. In case of an agent's action it is often dictated by the policy followed by the agent. For environmental actions, it encodes the various possibilities that are being accounted for in the model. Together the four-some $(\mathcal{S}, \mathcal{A}, \Phi, poss)$ are referred to as a system. The evolution of the world with respect to a system is characterized by the following definition.

Definition 1 An alternating sequence of states and actions $s_0, a_1, s_1, a_2, \ldots, s_k, a_{k+1}, s_{k+1}, \ldots$ is said to be a *trajectory consistent with a system* $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ if :

- $s_{k+1} \in \Phi(s_k, a_{k+1})$, and
- $a_{k+1} \in poss(s_k)$. □

A common restriction on how the world evolves is defined using the notion of *stability*. The following definition of stability is adapted from [OWA91] and has its origin in control theory and discrete event dynamic systems [OWA91,PB98,RW87b,RW87a].

Definition 2 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a set of states E, a state s is said to be *stable* in A w.r.t. E if all trajectories consistent with A and starting from s go through a state in E in a finite number of transitions and they visit E infinitely often afterwards. A set of states S is stable with respect to E if all states in S are stable with respect to E.

We say $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ is a stable system if all states in $\mathcal{S}$ are stable in A with respect to E. $\square$

Although the above definition of stability is with respect to a state E, it can be easily adapted to a formula φ. In that case E is the set of states where φ is $true$.

An alternative approach to characterize the evolution of states is through temporal operators. Some of the important future temporal operators, as discussed in [MP92], are: Next ($\bigcirc$), Always ($\Box$), Eventually ($\Diamond$), and Until ($\mathcal{U}$). Their meaning with respect a trajectory σ given by $s_0, a_1, s_1, \ldots, s_k, a_{k+1}, s_{k+1}, \ldots$ is defined as follows:

- $(\sigma, j) \models p$ iff p is true in s_j.
- $(\sigma, j) \models \bigcirc p$ iff $(\sigma, j+1) \models p$
- $(\sigma, j) \models \Box p$ iff $(\sigma, k) \models p$, for all $k \geq j$.
- $(\sigma, j) \models \Diamond p$ iff $(\sigma, k) \models p$, for some $k \geq j$.
- $(\sigma, j) \models p \; \mathcal{U} \; q$ iff there exists $k \geq j$ such that $(\sigma, k) \models q$ and for all i, $j \leq i < k$, $(\sigma, i) \models p$.

An alternative definition of stability which is equivalent to Definition 2 for finite $\mathcal{S}$ can be given as follows:

Definition 3 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a formula φ, let E be the set of states where φ is true. A state s is then said to be *stable* in A w.r.t. E if for all trajectories σ of the form $s = s_0, a_1, s_1, \ldots, s_k, a_{k+1}, s_{k+1}, \ldots$ consistent with A, $(\sigma, 0) \models \Box\Diamond\varphi$. $\square$

The advantage of using temporal operators, as in the the above definition, instead of Definition 2 is that the former allows us to specify a larger class of goals and build on top of the notion of stability. For example, a notion similar to stability, referred to as a *response property* [MP92] is of the form $\Box(p \rightarrow \Diamond q)$.

2.1 Stabilizability

The notion of stability is defined with respect to a system and the evolution of the world consistent with the system. When we focus on the agent and its ability to make a system stable, we need a notion of *stabilizability* which intuitively means that there exists a control policy which the agent can use to fashion a stable system.

Given a system $(\mathcal{S}, \mathcal{A}, \Phi, poss)$, when discussing stabilizability of the system, we need to consider the following additional aspects:

- $\mathcal{A}_{agent}$: the set of actions in $\mathcal{A}$ that the agent can execute.
- $exo : \mathcal{S} \rightarrow \mathcal{A}$, where $exo(s) \subseteq poss(s)$ is the set of exogenous actions that may occur in the state s.

Intuitively, given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, $\mathcal{A}_{agent}$, exo, and E, we say a state s is stabilizable with respect to E, if there exists a control function K such that

- $K(s) \in \mathcal{A}_{agent}$ and $K(s) \in poss(s)$,
- s is stable with respect to E in the system $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$, where $poss_{K,exo}(s) = K(s) \cup exo(s)$,
- and we do not reach a state where no actions are possible.

The last condition is referred to as s being alive. It is formally defined by the following two definitions, the first of which defines the set $R(A, s)$ of states that can be reached from s in the system A.

Definition 4 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a state s, $R(A, s)$ is the smallest set of states that satisfy the following conditions:

- $s \in R(A, s)$,
- If $s' \in R(A, s)$, and $a \in poss(s')$, then $\Phi(s', a) \subseteq R(A, s)$. □

Definition 5 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a state s, we say s is alive if $poss(s') \neq \emptyset$, for all $s' \in R(A, s)$. We say $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ is alive if all states in $\mathcal{S}$ are alive. □

We are now ready to formally define the notion of stabilizability.

Definition 6 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a function exo, and a set of states E, we say that $s \in \mathcal{S}$ is *stabilizable* with respect to E if there exists a control function K—that satisfies the conditions $K(s) \in \mathcal{A}_{agent}$ and $K(s) \in poss(s)$, such that s is alive and stable with respect to E in the system $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$.

A set of states S is stabilizable with respect to E if for all states s in S, s is stabilizable with respect to E due to a common control function. □

3 Running example: two finite buffers

In this section we introduce a running example which we will use in illustrating the notion of stabilizability and also other concepts through the rest of the paper. We imagine a system with two finite buffers, b_1 and b_2, where objects are added to b_1 in an uncontrollable way. An agent then moves objects from b_1 to b_2 and processes them there. When an object has been processed it is automatically removed from b_2. This is a slight modification of a finite buffer example from [PB98] and generalizes problems such as ftp agents maintaining a clean ftp area by moving submitted files to other directories, or robots moving physical objects from one location to another.

For simplicity, we assume that the agent has three control actions $\mathbf{M}_{12}$ that moves an object from b_1 to b_2 (if such an object exists), the opposite action, $\mathbf{M}_{21}$ that moves an object from b_2 to b_1, and **Pro** that processes and removes an object in b_2. There is one exogenous action, Ins that inserts an object into buffer b_1. The capacities of b_1 and b_2 are assumed to be equal.

If the control goal of this system is to keep b_1 empty, the system is not stabilizable, since objects can be continually inserted before the agent has a chance to empty the buffer. However, if no insertions are performed for a certain window of non-interference, the agent can always empty b_1. This implies that the system is maintainable but not stabilizable. We now make the above argument explicit by using an example.

Example 1 Running Example
We assume that the maximum capacity of the buffers is 3, and model the state space by letting every state in $\mathcal{S}$ represent the number of objects in b_1 and b_2. I.e., a state is identified by a pair of integers $\langle i, j\rangle$ (ij for short) where i denotes the number of objects in b_1 and j the number of objects in b_2. With the maximum capacity assumed to be 3, the state space consists of $4 \times 4 = 16$ states. That is

$$\mathcal{S}_b = \{0, 1, 2, 3\} \times \{0, 1, 2, 3\}.$$

The set of actions is $\mathcal{A}_b = \{\mathbf{M}_{12}, \mathbf{M}_{21}, \mathbf{Pro}, Ins\}$.

We assume that the transition function is deterministic (that is $\Phi_b : \mathcal{S}_b \times \mathcal{A}_b \to \mathcal{S}_b$) and is defined as follows:

$$\Phi_b(\langle i, j\rangle, \mathbf{M}_{12}) = \langle i - 1, j + 1\rangle$$
$$\Phi_b(\langle i, j\rangle, \mathbf{M}_{21}) = \langle i + 1, j - 1\rangle$$
$$\Phi_b(\langle i, j\rangle, \mathbf{Pro}) = \langle i, j - 1\rangle$$

$\Phi_b(\langle i,j \rangle, Ins) = \langle i+1, j \rangle$

The enabling function, $poss_b$, is defined as

$$\begin{array}{lll} \mathbf{M}_{12} \in poss_b(\langle i,j\rangle) & \text{iff} & i \geq 1 \text{ and } j \leq 2 \\ \mathbf{M}_{21} \in poss_b(\langle i,j\rangle) & \text{iff} & i \leq 2 \text{ and } j \geq 1 \\ \mathbf{Pro} \in poss_b(\langle i,j\rangle) & \text{iff} & j \geq 1 \\ Ins \in poss_b(\langle i,j\rangle) & \text{iff} & i \leq 2 \end{array}$$

The system A_b is then defined as $A_b = (\mathcal{S}_b, \mathcal{A}_b, \Phi_b, poss_b)$.

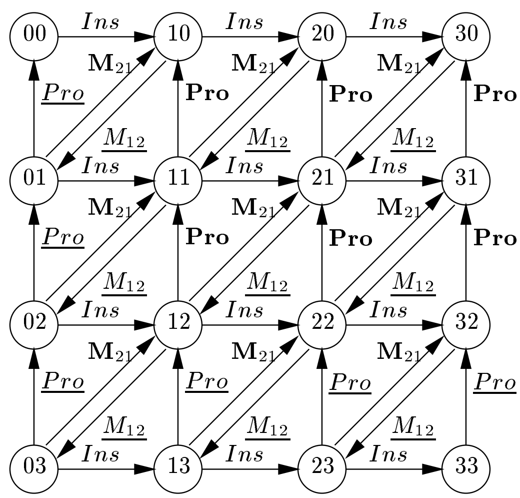

Fig. 1. The transition diagram of the system A_b for the buffer example.

It is easy to see that for $S = \{00\}$ (no objects in the buffers) and $E = \{00, 01, 02, 03\}$ (that is, we want to keep b_1 empty) S is not stabilizable w.r.t. E, since the exogenous action Ins can always interfere in the task of bringing the system back to E. For example, consider the control K_b defined as follows:

$K_b(\langle i,j \rangle) = \mathbf{M}_{12}$ when $i \geq 1$ and $j < 3$ and
$K_b(\langle i,j \rangle) = Pro$ when $(i = 0$ and $j \geq 1)$ or $(i \geq 1$ and $j = 3)$.

Intuitively, the above control directs the transfer of objects from buffer 1 to 2 whenever possible, and if that is not possible it then directs processing of objects in buffer 2 if that is possible. In Figure 1 the transitions due to K_b are marked with $\underline{M_{12}}$ and $\underline{Pro}$.

Consider the following trajectory consistent with the system $(\mathcal{S}_b, \mathcal{A}_b, \Phi_b, poss_{b_{K_b, exo}})$.

$00, Ins, 10, Ins, 20, Ins, 30, \mathbf{M}_{12}, 21, Ins, 31, Pro, \overline{30, \mathbf{M}_{12}, 21, Ins, 31, Pro}.$

In the above trajectory a state in E is never reached after the starting state. Similar trajectories can be found for any control and hence S is not stabilizable w.r.t. E.

On the other hand, $S = \{00\}$ is stabilizable w.r.t. $E' = \{0, 1, 2\} \times \{0, 1, 2, 3\}$ (that is, we allow at most two objects in b_1 at any time), since following K_b we can go from any of the states in $\{30, 31, 32, 33\}$ to E' with the execution of at most two control actions, while no exogenous actions are possible for those states. If we would introduce a failure state, for example for the case when b_1 is full and Ins is executed, S would no longer be stabilizable w.r.t. E'. In that case A_b would be only trivially stabilizable. □

4 K-maintainability

As we mentioned in Section 1 our main intuition behind the notion of maintainability is that maintenance becomes possible only if there is a window of non-interference from the environment during which the maintenance is done by the agent. When the agent needs a minimum window of k uninterfered steps we have a notion of k-maintenance. In other words an agent k-maintains a condition c if its control (or its reaction) is such that if we allow it to make the controlling actions without interference from the environment for at least k steps then it gets to a state that satisfies c. A question that may come to the mind of the reader at this point is: In presence of this window of non-interference from exogenous actions, what is the role of the exogenous actions or the environment?

We answer it as follows. Our definition of maintainability has the parameters as: (i) a set of initial states S that the system may be initially in, (ii) a set of desired states E that we want to maintain, (iii) a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, (iv) a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, (v) a function exo such that $exo(s) \subseteq poss(s)$, and (vi) a control law K such that $K(s) \in \mathcal{A}_{agent}$ and $K(s) \in poss(s)$. Our goal is to formulate when the control law K maintains E assuming that the system is initially in one of the states in S. We account for the exogenous actions by defining the notion—$Closure(S, A_{K,exo})$—of a closure of S with respect to $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$. This closure is the set of states that the system may get into starting from S because of K and/or exo. Then we define maintainability by requiring that the control law be such that if the system is in any state in the closure and is given a window of non-interference from exogenous actions then it gets into a desired state.

Now a question might be that suppose the above condition of maintainability is satisfied, and while the control law is leading the system towards a desired state an exogenous action happens and takes the system off that path. What then? The answer is that the state that the system will reach after the exogenous action will be a state from the closure. Thus, if the system is then left alone (without interference from exogenous actions) it will be again on its way to a desired state. So in our notion of maintainability, the control is always taking the system towards a desired state, and after any disturbance from an exogenous action, the control again puts the system back on a path to a desired state.

We now formally define the notions of closure and maintainability.

Definition 7 Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system and S be a set of states. By $Closure(S, A)$ (the closure of A w.r.t. S) we refer to the set $\bigcup_{s \in S} R(A, s)$. $\square$

Following are some of the properties of the set $Closure(S, A)$, which trivially follows from the definition of $R(A, s)$.

Lemma 1 Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system and S be a set of states.

(i) $S \subseteq Closure(S, A)$.
(ii) If $s \in Closure(S, A)$, and $a \in poss(s)$, then $\Phi(s, a) \subseteq Closure(S, A)$. $\square$

Definition 8 Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system, s be a state, E be a set of states and K be a control. $Unfold_k(s, A, K, E)$ is the set of sequences $s = s_0, s_1, \ldots, s_k$ where $s_{j+1} \in \Phi(s_j, K(s_j))$. $\square$

Intuitively, an element of $Unfold_k(s, A, K, E)$ is a sequence of states of length $k+1$ that the system may go through if it follows the control K starting from the state s.

Definition 9 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a specification of exogenous action occurrence exo, and a set of states E, we say a set of states S is *k-maintainable* with respect to E if there exists a control law K such that for each state s in $Closure(S, A_{K,exo})$, and for each sequence $s = s_0, s_1, \ldots, s_k$ in $Unfold_k(s, A, K, E)$, $\{s_0, \ldots, s_k\}$ intersects with E.

If there exists an integer n such that S is *n-maintainable* with respect to E, we say S is *maintainable* with respect to E.

If $S = \mathcal{S}$, then we say A is maintainable with respect to E. $\square$

Intuitively, for a sequences $s = s_0, s_1 \ldots, s_k$ in $Unfold_k(s, A, K, E)$, the set of states $\{s_0, s_1, \ldots, s_k\}$ intersects with E means that we can get to a state in E from s with at most k transitions—where each transition is dictated by the control K—if the world were to unfold as in $s = s_0, s_1, \ldots, s_k$.

Example 2 Running Example, Cont'd
Earlier we showed that in A_b, $S = \{00\}$ is not stabilizable w.r.t. $E = \{00, 01, 02, 03\}$. Is then S maintainable w.r.t. E? Yes, it is. For the worst case system state, 33, a control law can move the system to 30 (by three transitions due to **Pos**) without the risk of interfering occurrences of exogenous actions. If there then is three transitions without interference the control law can apply $\mathbf{M}_{12}$ three times and be in 03. This implies that S is 6-maintainable w.r.t. E. We can with a similar argument show that S is 9-maintainable w.r.t. $\{00\}$. Similar argument can be made with respect to the control K_b of Example 1.

However, we have that S is *not* maintainable w.r.t., for example, $\{03\}$ (Since we cannot go from, for example, $\{00\}$, to $\{03\}$ with control actions only). $\square$

As the above example points out it is possible that S is maintainable with respect to E but not stabilizable with respect to it. The converse is also possible. In other words in certain cases we may have a system where a given S is stabilizable with respect to a set E, but yet is not maintainable. This happens when all paths between a state in S and a state in E has at least one exogenous action. In that case the agent, who does not have control over the exogenous actions, can not on its own make the transition from a state in S to a state in E. In a later section (Section 6) we present a more general notion of (k,l)-maintainability which generalizes both maintainability and stabilizability.

4.1 An alternative characterization of k-maintainability

The characterization of stability and stablizabilitry in Section 2 is based on putting conditions on trajectories obtained from the transition graph of a system. Such a characterization has the advantage that it is amenable to developing temporal operators that can express more general conditions. The definition of maintainability in Definition 9 is not based on trajectories. In this sub-section we give an alternative characterization based on trajectories and later in Section 7 we propose matching temporal operators.

Proposition 1 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a specification of exogenous action occurrence exo, and a set of states E, a set of states S is *k-maintainable* with respect to E iff there exists a control law K such that for each state s in S, and for all trajectories of the form $s = s_0, a_1, s_1, a_2, \ldots, a_j, s_j, a_{j+1}, \ldots$ that is consistent with $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$ if there is an i such that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$, then $\{s_{i+1}, \ldots, s_{i+k}\} \cap E \neq \emptyset$. $\square$

For the alternative characterization implied by the above proposition it is often more convenient to define a notion of 2-systems which makes the difference between agent's actions and exogenous actions explicit.

Definition 10 A 2-system B is a 5-tuple $(\mathcal{S}, \mathcal{A}_{agent}, \mathcal{A}_{exo}, \Phi, poss)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}_{agent}$ is a finite set of control actions, $\mathcal{A}_{exo}$ is a finite set of exogenous actions, Φ is a function from $\mathcal{S}$ to $2^{\mathcal{A}_{agent} \cup \mathcal{A}_{exo}}$ listing what actions and events may occur (or are executable) in what state, and $poss$ is a transition function from $\mathcal{S}$ and $\mathcal{A}_{agent} \cup \mathcal{A}_{exo}$ to $2^{\mathcal{S}}$. □

5 Algorithms for verifying maintainability and constructing controls that maintain

In this section we present a simple algorithm to verify the maintainabilty of control and give an algorithm that can (sometimes) construct a control that maintains.

5.1 Testing maintainability

Before presenting or algorithm for testing maintainability we first discuss some notations that we will use. With $Closure(S, A_{K,exo})$ and a set of states E we associate the set $Seq(S, A_{K,exo}, E)$ of sequences $s = s_0, s_1, \ldots, s_{|\mathcal{S}|}$, one for each $s \in Closure(S, A_{K,exo})$, where $s_{k+1} = s_k$ if $s_k \in E$, and $s_{k+1} \in \Phi(s_k, K(s_k))$ otherwise. Intuitively, $Seq(S, A_{K,exo}, E)$ is a set of sequences representing how the world evolves—starting from each state in $Closure(S, A_{K,exo})$—for the next $|\mathcal{S}|$ steps as forced by the control K and with the assumption that no exogenous action take place. Note that since we allow non-deterministic actions there may be multiple sequences with the same initial state. Since the control is supposed to maintain E, once a state in E is reached in the sequence, the subsequent states are repeatations of that state. In addition we only consider $|\mathcal{S}|$ steps because if a state in E is not reached within that many steps it means that some states were reached more than once and a state in E may never be reached.

Using $Seq(S, A_{K,exo}, E)$ we also define

$$Seq_{\cup}(S, A_{K,exo}, E) = \{\{s_0, \ldots, s_{|\mathcal{S}|}\} \mid s_0, \ldots, s_{|\mathcal{S}|} \in Seq(S, A_{K,exo}, E)\}.$$

We now present the maintainability testing algorithm.

Input: A system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of states E, a set of states S, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a function exo, and a control K.

Output: To find out if S is maintainable with respect to E, using the control K.

Algorithm:

Step 1: Compute $Closure(S, A_{K,exo})$ as follows:

(i) $S_{in} := S$. $S_{out} = \emptyset$.
(ii) While $S_{in} \neq S_{out}$ Do.
  (a) Pick an s from $S_{in} \setminus S_{out}$.
  (b) $S_{in} := S_{in} \cup \Phi(s, K(s)) \cup \{s' : s' \in \Phi(s, b),$ for some $b \in exo(s)\}$
  (c) $S_{out} := S_{out} \cup \{s\}$.

Step 2: Compute $Seq(S, A_{K,exo}, E)$.

Step 3: If for all $Y \in Seq_{\cup}(S, A_{K,exo}, E)$, $Y \cap E \neq \emptyset$ then S is maintainable with respect to E, using the control K; Otherwise it is not maintainable with respect to E, using the control K. □

Proposition 2 Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system, E be a set of states, and K be a control. For every $Y \in Seq_{\cup}(S, A_K, E)$, $Y \cap E \neq \emptyset$ iff S is maintainable w.r.t. E, using the control law K. □

Proof: ($\Rightarrow$) Straightforward, since the control law can move the system from any state in $Closure(S, A_{K,exo})$ to a state in E in at most $|\mathcal{S}|$ transitions.

($\Leftarrow$) If S is maintainable w.r.t. E using K, there is a natural number n such that from every state in $Closure(S, A_{K,exo})$ there exist a sequence of states ending in E of length at most n, where every transition between states is due to K. Clearly, $n \leq |\mathcal{S}|$ so we repeat the last state in all sequences so that all sequences are of length $|\mathcal{S}|$ and gather all such sequences into one set. This set is clearly equal to $Seq(S, A_{K,exo}, E)$. Since every such sequence contains a member of E the proposition holds. □

The complexity of this algorithm depends on the computations in steps 1 and 2. Let $n = |\mathcal{S}|$ and $S = \mathcal{S}$ (this maximizes $Closure(S, A_{K,exo})$ and is, thus, the worst case). Clearly, we can compute $R(A_{K,exo}, s)$, for every $s \in S$, in polynomial time (e.g. by building the depth-first search tree, and simply collecting the vertices) with respect to n, and since we only need to to this n times, the computation is polynomial in n. In the same manner we can compute $Seq(S, A_{K,exo}, E)$ by recording the sequences that reach E during the depth-first search.

5.2 Control generation algorithm

In this subsection we present a sound algorithm that generates a maintainable control. (This algorithm is an adaptation of a similar algorithm from [BS98].) For that we first need to define the notion of a plan from a state to a set of states. In our definition of a *plan* we extend the function Φ to be applicable to sequence of actions as follows: For a sequence of actions $A = [a_0, a_1, \ldots, a_n]$ we define $\Phi(s, A)$

$$\Phi(s, [a_0, a_1, \ldots, a_n]) = \bigcup_{s' \in \Phi(s, a_0)} \Phi(s', [a_1, \ldots, a_n]).$$

Definition 11 Let A be a system, s be a state, and E be a set of states. We call the sequence of control actions $a_1, \ldots, a_n$ a *plan from s to E* iff $\Phi(s, [a_1, \ldots, a_n]) \subseteq E$. □

Our control generation algorithm will use calls to a planner that can plan with nondeterministic transition functions as a subroutine. Although, most planning algorithms assume that Φ is deterministic, the subtleties involved in planning when Φ is non-deterministic (referred to as nondeterministic planning) is discussed in [GT99].

Input: A system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of states E, a set of states S, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a function *exo*.

Output: A control K such that S is maintainable with respect to E, using the control K.

Algorithm:

Step 0: $S_{in} := S$, $S_{out} = \emptyset$.

Step 1: While $S_{in} \neq S_{out}$ Do.

(i) Pick an s from $S_{in} \setminus S_{out}$.
(ii) Find a *plan of minimal length* from s to a state in E using only control actions.
(iii) If no such plan exists then EXIT and return(FAIL).
(iv) Otherwise, let a be the first action of that plan.
  (a) Assign $K(s) = a$.
  (b) $S_{out} := S_{out} \cup \{s\}$.
  (c) $S_{in} := S_{in} \cup \Phi(s, a) \cup \{y : y \in \Phi(x, b)$, for some $b \in exo(s)\}$.

Step 2: If $S_{in} = S_{out}$, return(S_{out}, K). $\square$

Proposition 3 (Soundness) If the above algorithm terminates by returning S' and K, then: (i) $S' = Closure(S, A_{K,exo})$, and (ii) S is maintainable with respect to E, using the control K. $\square$

Proof:
We assume that for a particular S the algorithm has returned S' and K.

(i) We start by showing that $S_{out} \subseteq Closure(S, A_{K,exo})$ is an invariant of the algorithm by induction over the number of iterations, n, of the algorithm, and write S_{out}^n and S_{in}^n to denote the corresponding sets of states after n iterations. Trivially, $S_{out}^0 \subseteq Closure(S, A_{K,exo})$, and $S_{out}^1 \subseteq Closure(S, A_{K,exo})$ holds since a member of S_{out}^1 necessarily is a member of S. We assume that $S_{out}^k \subseteq Closure(S, A_{K,exo})$ and inspect a state $s \in S_{out}^{k+1}$ such that $s \notin S_{out}^k$. Again, if $s \in S$ we have $s \in Closure(S, A_{K,exo})$ by the definition of closure. Otherwise, there exists a state $s' \in S_{in}^k$ and an action a such that $s \in \Phi(s', a)$, from the construction of S_{in}^k. By the induction hypotheses we have that $s' \in Closure(S, A_{K,exo})$ and since s is reachable from s' we also have $s \in Closure(S, A_{K,exo})$.

For the other inclusion we take a state $s \in Closure(S, A_{K,exo})$. By the definition of closure there exists a state $s' \in S$ such that s is reachable from s' which implies that there is a trajectory consistent with $A_{K,exo}$: $s' = s_0, a_1, s_1, \ldots, a_n, s_n = s$ where $s_{k+1} \in \Phi(s_k, a_{k+1})$ and $a_{k+1} \in poss_{K,exo}(s_k)$. We prove that $s \in S'$ by induction over the length of the trajectory.

For $n = 0$ we have $s \in S \subseteq S'$. We assume that s_k belongs to S' for all $k < n$. By the definition of trajectories consistent with $A_{K,exo}$ we have that $a_{k+1} \in poss_{K,exo}(s_k)$ which implies either that $K(s_k) = a_{k+1}$ (1)
or $a_{k+1} \in exo(s_k)$ (2)
By the induction hypotheses, s_k is added to S_{out} at some iteration of the algorithm, and in the same iteration all members of the set $L = \Phi(s_k, a_{k+1}) \cup \{y : y \in \Phi(s, b), \text{ for some } b \in exo(s)\}$ are added to S_{in}. From (1) or (2) we get that s_{k+1} belongs to L, which means that it will be picked by the algorithm in a later iteration, and added to S_{out} and thus belongs to S'. Therefore, $S' = Closure(S, A_K)$.

(ii) From (i) we know that $S' = Closure(S, A_{K,exo})$, so we may construct $Seq(S, A_{K,exo}, E)$. Since the algorithm selects states reached by (sub-) plans of minimal length, the same state (except for the last that belongs to E) cannot occur more than once in every sequence in $Seq(S, A_{K,exo}, E)$. It is also clear that in every such sequence there is a state belonging to E. Thus, we can apply proposition 2, which proves this proposition. $\square$

The complexity of the above algorithm depends on the size of the closure and the time it takes to find minimal cost plans. The worst case size of the closure is the total number of states, say n. Assuming the agent's actions to be deterministic, the complexity of finding a minimal cost plan can be thought of as finding minimal cost paths in a graph, which using Dijkstra's algorithm is quadratic in the number of nodes in the graph. If we consider the function Φ to be encoded in a language such as $\mathcal{A}$ then planning is PSPACE-complete with respect to the number of fluents or state variables m. (Note that planning restricted to polynomial length plans is NP-complete when Φ is deterministic and Σ_2P-complete when Φ is non-deterministic. But by restricting to polynomial length plans we may miss plans.)

Our algorithm is not complete in the sense that it may not return a control that maintains while such a control may exist. A complete algorithm can be constructed by searching over the space of controls. The significance of our algorithm is its incrementality. Suppose new states are added to S, the set of initial states and S is updated to S'. In that case often we can just update the current control K by adding new rules without making any changes to the old rules. This issue is further discussed in a slightly different framework in [BS98].

At this point we would like to point out the relation between our work here and some research on reactive and situated agents [KR91]. In [KR91], they say that in a control rule 'if c then a', the action a, must be the action that *leads* to the goal from any situation that satisfies the condition c. The above algorithm interprets the notion of 'leading to' as the first action of a minimal length plan.

6 Generalization: (k, l)-Maintainability

In this section we generalize the notion of maintainability and show that the notion of stabilizability is a special case of this generalization. Our generalization is based on the intuition that perhaps, we can allow a limited number of exogenous actions during our so called 'window of non-interference' and still be able to get back to a state in E. Such a notion incorporates the possibility that it can take advantage of exogenous actions where certain transitions are beyond the direct capability of the agent. We refer to this general notion as (k, l)-*maintainability*.

Definition 12 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a specification of exogenous action occurrences exo, and a set of states E, we say a set of states S is (k, l)-*maintainable* ($l \leq k$) with respect to E if there exists a control law K such that for each state s in $Closure(S, A_{K,exo})$, all trajectories—consistent with $A_{K,exo}$—from s whose next k transitions contain at most l transitions due to exogenous actions and the rest is dictated by the control K, reach a state in E by the k-th transition. □

Proposition 4 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a specification of exogenous action occurrences exo, and a set of states E, a set of states S is (k,0)-maintainable iff it is k-maintainable. □

Proof: S is (k,0)-maintainable
iff there exists a control law K such that for each state s in $Closure(S, A_{K,exo})$, all trajectories—consistent with $A_{K,exo}$—from s whose next k transitions contain at most 0 transitions due to exogenous actions and the rest is dictated by the control K, reach a state in E by the k-th transition.
iff for each state s in $Closure(S, A_{K,exo})$ and for each sequence $s = s_0, s_1, \ldots, s_k$ in $Unfold_k(s, A, K, E)$, the set of states $\{s_0, s_1, \ldots, s_k\}$ intersects with E.
iff S is k-maintainable. □

Proposition 5 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agent's actions $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a specification of exogenous action occurrences exo, and a set of states E, a set of states S is stabilizable iff all states in S are alive and there exists an integer m such that S is *(m,m)-maintainable* with respect to E. □

Proof: ($\Rightarrow$) Suppose S is stabilizable with respect to E. That means there is a control function K such that S is alive and stable with respect to E in the system $A_{K,exo}$. That means all trajectories consistent with $A_{K,exo}$ and starting from any state s in S go through a state in E in a finite number of transitions and they visit E infinitely often afterwards.

Case 1. Let $s \in S$. Then by assumption all trajectories from s go through a state in E within finite number of transition. Since there are a finite number of such trajectories, by n_s let us denote the maximum number of transition to go from s to a state in E, among the different trajectories. This includes some actions due to exogenous actions, but at most n_s of them.

Case 2. If $s \in Closure(S, A_{K,exo}) \setminus S$, then the state s is in a trajectory T from a state, say s', in S. Since all trajectories consistent with $A_{K,exo}$ reach from a state s' in S to a state in E within finite number of transitions and visit E infinitely often afterwards, this partial trajectory T' of T which starts from the state s will visit E infinitely often. So through this trajectory T', we can reach from s to a state in E within finite number of transitions. Let n_s me the maximum number over all such trajectories. This includes at most n_s transitions which are due to exogenous actions.

Let $m = max\{n_s | s \in Closure(S, A_{K,exo})\}$. Since $Closure(S, A_{K,exo})$ is finite, m exists and is an integer. Now from each state s in $Closure(S, A_{K,exo})$, all trajectories, consistent with $A_{K,exo}$, whose next m transitions contain at most

m transitions due to exogenous actions and the rest is dictated by the control K reach a state in E within m transitions. Thus S is (m, m)-maintainable with respect to E. $\Box$.

$(\Leftarrow)$ Suppose that S is alive and there exists an integer m such that S is (m, m)-maintainable with respect to E. Thus there exists a control law K such that from each state s in $Closure(S, A_{K,exo})$, all trajectories from s consistent with $A_{K,exo}$ whose next m actions contain at most m transitions due to exogenous actions reach a state in E within m transition. Thus all trajectories starting from s and consistent with $A_{K,exo}$ reach a state s' in E within finite number of transitions. Any state s'' which we can reach from that state s' belongs to $Closure(S, A_{K,exo})$, as that state s'' can be reached from s. Thus all trajectories from s'' which are consistent with $A_{K,exo}$ whose next m actions contain at most m transitions due to exogenous actions reach a state in E within m transitions again. Hence all trajectories from $s \in Closure(S, A_{K,exo})$ consistent with $A_{K,exo}$ visit E within the next m transitions every time they leave E to a state outside of E. I.e. they visit E infinitely often afterwards. Thus S is stable with respect to E in the closed loop system $A_{K,exo}$. Since S is alive by assumption, S is stabilizable with respect to E. $\Box$

7 Conclusion, related work and future directions

In this paper we give a formal characterization of maintenance goals and distinguish it from the notions of stabilizability and temporal goals of the form $\Box\Diamond f$. We present several motivating examples that illustrate the need for our notion of maintainability. The basic idea being that for certain kinds of maintenance it is important that the maintaining agent be given a window of non-interference from the environment so that it can do the maintenance. To formalize this we need to need to distinguish between the agent's actions and the environment's actions.

In our formalization we define the notion of k-maintainability, where k refers to the maximum window of opportunity necessary for the maintenance. We show that the notions of k-maintainability and stabilizability are incomparable and introduce a notion of (k,l)-maintainability that generalizes both. We present a simple algorithm for verifying maintainability, and also present an algorithm that can sometimes construct a maintainable control.

Besides the related works we already mentioned such as stabilizability and temporal logic, the notion of maintenance has appeared in AI in many other papers. For example, in [Ort99], Ortiz discusses maintenance actions. His notion of maintenance is stronger than both the notion of stabilizability and our notion as he requires the formula that is maintained to be true throughout. The notion of maintenance is also related to the notion of 'execution monitoring' which is studied in the context of robot programs in [DRS98]. In 'execution monitoring' the world is monitored and if a discrepancy is found between the prediction made by the agent and the real world, then new plans are made to recover from the discrepancy. A deliberative architecture for maintenance can be extrapolated from the notions in [BGP97], where an agent executes a cycle of $observe; assimilate; (re)plan\_from\_current\_situation; execute\_part\_of\_the\_plan$.

Outside of AI, our notion of maintenance is very closely related to the notion of self-stabilization in [Dij74] which is used in characterizing fault-tolerant systems. There the environmental actions are referred to as transient failures and unlike in our case are never beneficial. Some of the other differences between their approach and ours are: (i) Our agents and environments include physical domains, while theirs is about the behavior of a computer program; (ii) our actions can be non-deterministic while their program statements are deterministic; (iii) their control is not allowed to take the environment outside of E from inside of E while we do not have such a restriction ; and (iv) they do not have a notion similar to our (k,l)-maintainability.

Among the future directions of this work, we need to further study algorithms for computing maintainable controls and find the lower bound of the complexity of determining the existence of a maintainable control. We are currently working on this. Along those lines, one approach is not to have S as an input but rather start from E and find the largest S from which E is k-maintainable. Such an approach towards stabilizability is taken in [OWA91].

Another future direction is to develop temporal operators that can specify maintainability. The advantage of having such temporal operators and a temporal logic based on it is that we can use the logic to construct more involved specifications of agents. For example we may have switches f_1, f_2 and depending on their initial value the agent may be required to maintain one of the formulas ϕ_0, ϕ_1, ϕ_2, and ϕ_3. Along these lines we have the following definition of some new temporal operators, and we plan to further study logics based on such operators.

- $(\sigma, j) \models \bigcirc_k p$ iff $i \geq j$ is the smallest index such that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$ and $(\sigma, i + r) \models p$, for some $1 \leq r \leq k$.
  Intuitively, $(\sigma, j) \models \bigcirc_k p$ means that starting from s_j, the next time there is a sequence of k consecutive transitions due to the agent, p holds in one of the states in that sequence.
- $(\sigma, j) \models \Box_k p$ iff for all $i \geq j$ if $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$ then $(\sigma, i + r) \models p$ for some $1 \leq r \leq k$.
  Intuitively, $(\sigma, j) \models \Box_k p$ means that starting from s_j, within or after any k consecutive transitions due to control actions p holds.

- $(\sigma, j) \models \bigcirc_{k,l} p$ iff $i \geq j$ is the smallest index such that $|\{a_{i+1}, \ldots, a_{i+k}\} \cap \mathcal{A}_{exo}| \leq l$ and $(\sigma, i+r) \models p$ for some $1 \leq r \leq k$. Intuitively, $(\sigma, j) \models \bigcirc_{k,l} p$ means that starting from s_j, the next time there is a sequence of k consecutive actions with at most l environmental actions, p holds in one of the states in that sequence.
- $(\sigma, j) \models \Box_{k,l} p$ iff for all $i \geq j$ if $|\{a_{i+1}, \ldots, a_{i+k}\} \cap \mathcal{A}_{exo}| \leq l$ then $(\sigma, i+r) \models p$ for some $1 \leq r \leq k$.
  Intuitively, $(\sigma, j) \models \Box_{k,l} p$ means that starting from x_j, within or after any k transitions with at most l exogenous actions p holds.

Some examples of specifications using the above operators are as follows:

- After the bell is pressed (b) service is provided (p) within the next window of opportunity of length l.
  $b \rightarrow \bigcirc_l p$
- Based on the value of the switches f_1 and f_2 to $\phi_0, \phi_1, \phi_2, \phi_3$ has to be k-maintained respectively.
  $(\neg f_1 \wedge \neg f_2 \Rightarrow \Box_k \phi_0) \vee (\neg f_1 \wedge f_2 \Rightarrow \Box_k \phi_1) \vee (f_1 \wedge \neg f_2 \Rightarrow \Box_k \phi_2) \vee (f_1 \wedge f_2 \Rightarrow \Box_k \phi_3))$

References

[BGP97] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, May 1997.

[BK98] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Annals of Math and AI*, 22:5–27, 1998.

[BKT01] C. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning with temporal goals. In *IJCAI 2001*, 2001.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, pages 14–23, April 1986.

[BS98] C. Baral and T. Son. Relating theories of actions and reactive control. *Electronic transactions on Artificial Intelligence*, 2(3-4), July-Dec 1998.

[CES86] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CW90] S. Ceri and J. Widom. Deriving production rules for constraint maintainance. In *VLDB 90*, pages 566–577, 1990.

[Dij74] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *CACM*, 17(11):843–644, November 1974.

[DRS98] G. DeGiacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proc. of KR 98*, pages 453–464. Morgan Kaufmann, 1998.

[FN71] R. Fikes and N. Nilson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[GL92] M. Gelfond and V. Lifschitz. Representing actions in extended logic programs. In K. Apt, editor, *Joint International Conference and Symposium on Logic Programming.*, pages 559–573. MIT Press, 1992.

[GT99] F. Giunchiglia and P. Traverso. Planning as model checking. In S. Biundo and M. Fox, editors, *Recent Advances in AI Planning : Proceedings of the 5th European Conference on Planning, ECP'99*, 1999.

[KBSD97] F. Kabanza, M. Barbeau, and R. St-Denis. Planning control rules for reactive agents. *Artificial Intelligence*, 5(1):67–113, 1997.

[KR91] L. Kaelbling and S. Rosenschein. Action and planning in embedded agents. In P. Maes, editor, *Designing Autonomous Agents*, pages 35–48. MIT Press, 1991.

[Mae91] P. Maes, editor. *Designing Autonomous Agents.* MIT/Elsevier, 1991.

[MP92] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: specification.* Springer Verlag, 1992.

[NB00] M. Nakamura and C. Baral. Invariance, maintenance and other declarative objectives of triggers – a formal characterization of active databases. In *Proc. of First International Conference on Computational Logic (DOOD track) CL'2000*, pages 1210–1224, 2000.

[NS00] R. Niyogi and S. Sarkar. Logical specification of goals. In *Proc. of 3rd international conference on Information Technology*, pages 77–82, 2000.

[Ort99] C. Ortiz. A commonsense language for reasoning about causation and rational action. *Artificial Intelligence*, 111(2):73–130, 1999.

[OWA91] O. Ozveren, A. Willsky, and P. Antsaklis. Stability and stabilizability of discrete event dynamic systems. *JACM*, 38(3):730–752, July 1991.

[PB98] K. Passino and K. Burgess. *Stability Analysis of Discrete Event Systems.* Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley and Sons, Inc., New York, 1998.

[Rei01] R. Reiter. *Knowledge in action: logical foundation for describing and implementing dynamical systems.* MIT press, 2001.

[RW87a] P. Ramadge and W. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1217, September 1987.

[RW87b] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event process. *SIAM Journal of Control and Optimization*, 25(1):206–230, Jan 1987.

[Son99] E. Sontag. Stability and stabilization: Discontinuities and the effect of disturbances. In F. Clarke and R. Stern, editors, *Proc. NATO advanced study institute, July/Aug 1998*, pages 551–598. Kluwer, 1999.

[WC96] J. Widom and S Ceri, editors. *Active Database Systems - Triggers and Rules for advanced database processing.* Morgan Kaufmann, 1996.

[WE94] D. Weld and O. Etzioni. The first law of robotics (a call to arms). In *AAAI*, pages 1042–1047, 1994.