Maintainability: a weaker stabilizability like notion for high level control

Mutsumi Nakamura
Department of CSE
University of Texas at Arlington
Arlington, TX 76019, USA
nakamura@cse.uta.edu

Chitta Baral
Department of CSE
Arizona State University
Tempe, AZ 85287, USA
chitta@asu.edu

Marcus Bjäreland
Department of Comp and Info Sc
Linköping University
S-581 83 Linkoping, Sweden
marbj@ida.liu.se

Abstract

The goal of most agents is not just to reach a goal state, but rather also (or alternatively) to put restrictions on its trajectory, in terms of states it must avoid and goals that it must 'maintain'. This is analogous to the notions of 'safety' and 'stability' in the discrete event systems and temporal logic community.

In this paper we argue that the notion of 'stability' is too strong for formulating 'maintenance' goals of an agent – in particular, reactive and *software agents*, and give examples of such agents. We present a weaker notion of 'maintainability' and show that our agents which do not satisfy the stability criteria, do satisfy the weaker criteria. We give algorithms to test maintainability, and also to generate control for maintainability. We then develop the notion of 'supportability' that generalizes both 'maintainability' and 'stabilizability, develop an automata theory that distinguishes between exogenous and control actions, and develop a temporal logic based on it.

Motivation and Introduction

Stability has undergone extensive investigations in the control theory community (Passino & Burgess 1998), both for continuous systems (e.g. Lyapunov stability and asymptotic stability) and Discrete Event Dynamic Systems (DEDS) (Ramadge & Wonham 1987b; 1987a; Ozveren, Willsky, & Antsaklis 1991). All these notions can be summarized as in (Passino & Burgess 1998):

> We say that a system is stable if when it begins in a good state and is perturbed into any other state it will always return to a good state.

The appropriate stability notion in a particular case depends on how the notions "system", "begins", "state", "good", and "perturbed" are defined, For DEDS the mainstream definition can be found in (Ozveren, Willsky, & Antsaklis 1991), and that definition is the one we use in this paper. They also mention that relation between stability and the notions of safety, fairness, livelock, deadlock are well studied. *In this paper we present a related notion which we call* **maintainability**, *and argue its importance, particularly for high level control of agents.*

Copyright © 2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Intuitively, we can view stabilizability as a hard constraint of the system while maintainability is a softer constraint. In both maintainability and stabilizability our goal is that the system should be among a given set of states E as much as possible. In stabilizability, we want a control such that regardless of where the system is now and what exogenous actions may happen, the system will reach one of the states in E within a finite number of transitions and keep visiting it infinitely often after that. In maintainability, we have a weaker requirement where the system reaches a state in E within a finite number of transitions, provided it is not interfered with during those transitions. Thus in maintainability, we admit that if there is continuous interference (by exogenous actions) we can not get to E in a finite number of transition. Such a system will not satisfy the condition of stabilizability, but may satisfy the condition of maintainability.

Many practical closed-loop systems are not stabilizable, but they still serve a purpose and we believe that such systems purpose can be specified by using the weaker notion of maintainability. An example of such a system is an active database system (Widom & Ceri 1996) where 'consistency' of data is 'maintained' using active rules (also referred to as triggers). In such a database system, external updates are made to the database through Insert, Delete and Update commands. But the direct result of the updates may take the database to an inconsistent state where 'integrity constraints' of the database may be violated. In that case, the active part of the database triggers rules that result in additional changes to the database to bring it back to a consistent state. Now suppose E is the set of consistent states of a database. We can not capture the correctness of the triggers by directly using the notions of 'stability'. That is because, if there is a continuous stream of external updates with no time in between for getting back to consistency, then there is no guarantee that the database will reach a state in E within a finite number of transitions. But we can have a different notion of correctness of triggers, where the triggers are correct if given a window of non-interference (from external updates) the triggers will ultimately make the database consistent. In fact that is what happens in a database system where external updates are blocked until the triggers bring back the database to a consistent state.

Another example is a mobile robot (Brooks 1986; Maes 1991) 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.

Other examples include agents that perform tasks based on commands. Here, the correctness of the agent's behavior can be formalized as 'maintaining' states where there are no commands in the queue. We can not use the notion of stability because if there is a continuous stream of commands, then there is no guarantee that the agent would get to a state with no commands in its queue within a finite number of transitions.

The rest of the paper is structured as follows. We first formally define the notion of stability and stabilizability. We then introduce the notion of maintainability and compare it with the notion of stabilizability. Next we show that the correctness of an active database can be formalized as maintainability of consistent states. We then present algorithms to verify maintainability, and to construct controls to make a system maintain a set of states. Finally, we develop a general notion called *supportability* and show that stabilizability and maintainability are special cases of it.

Reviewing stability and stabilizability

In this section we review the notions of stability and stabilizability adapted from the definitions in (Ozveren, Willsky, & Antsaklis 1991).

Stability and aliveness

Definition 0.1 A system A is a 4-tuple (X, Σ, f, d), where X is a finite set of states, Σ is a finite set of actions, d is a function from X to 2^{Σ} listing what actions may occur (or are executable) in what state, and f is a non-deterministic transition function from X and Σ to 2^X. $\Box$

Definition 0.2 A trajectory is an alternating sequence of states and actions, and could be either a finite trajectory that starts and ends with a state or an infinite trajectory.

A trajectory $x_0, a_1, x_1, a_2, \ldots, x_k, a_{k+1}, x_{k+1}(\ldots)$ is said to be consistent with a system A if:

- $x_{k+1} \in f(x_k, a_{k+1})$, and
- $a_{k+1} \in d(x_k)$. $\Box$

Definition 0.3 Given a system A and a set of states E, a state x is said to be *stable* in A w.r.t. E if all trajectories consistent with A and starting from x go through a state in E in a finite number of transitions and they visit E infinitely often afterwards.

We say $A = (X, \Sigma, f, d)$ is a stable system if all states in X are stable in A w.r.t. E. $\Box$

Alternatively, A is stable w.r.t. E if, for any state $x \notin E$, every infinite trajectory starting with x will lead to E in a finite number of steps.

Definition 0.4 $R(A, x)$ denotes the set of states that can be reached from x in a system A.

A state x is said to be *alive* if $d(y) \neq \emptyset$, for all $y \in R(A, x)$. (I.e., we can not reach a state y from x, where no action is possible.)

We say $A = (X, \Sigma, f, d)$ is *alive* if all states in X are alive. $\Box$

Stabilizability

We now consider control and exogenous actions. The set of control actions U is a subset of Σ, that can be performed by the (controlling) agent. A particular control K is a function from X to U. The set of exogenous actions that can occur in a state (and that are beyond the control of the agent) is given by a function e from X to 2^{Σ}, such that $e(x) \subseteq d(x)$.

Definition 0.5 Let $A = (X, \Sigma, f, d)$ be a system. In presence of e, U, and K, we define A_K[1] , the closed loop system of A as the four-tuple (X, Σ, f, d_K), where $d_K(x) = (d(X) \cap \{K(x)\}) \cup e(x)$. $\Box$.

Definition 0.6 Given a system A, a function e, and a set of states E, we say $S \subseteq X$ is *stabilizable* with respect to E if there exists a control law[2] K such that for all x in S, x is alive and stable with respect to E in the closed loop system A_k. If $S = X$, we say A is stabilizable with respect to E. $\Box$

Maintainability

Our intuition behind maintainability is that we would like our system to 'maintain' a formula (or a set of states where the formula is satisfied) in presence of exogenous actions. By 'maintain' we mean a weaker requirement than the temporal operator $always$ ($\Box$) where $\Box f$ means that f should be true in all the states in the trajectory. The weaker requirement is that our system needs to get to a desired state within a finite number of transitions provided it is not interfered in between by exogenous actions. The question then is what role the exogenous actions play.

Our definition of maintainability has parameters as a set of initial states S, that the system may be initially in, a set of desired state E, that we want to maintain, a system A and a control law K. 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 – $Closure(S, A)$ – of a closure of S with respect to A. This closure is the set of states that the system may get into starting from S. Then we define *maintainability* by requiring that the control law be such that

[1]A more appropriate terminology would be $A_{K,e}$. We use A_K to remain consistent with the usage in (Ozveren, Willsky, & Antsaklis 1991).

[2]It is also referred to as 'feedback law', 'feedback control' or 'state feedback' in the literature.

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 on a path to a desired state.

We now formally define the notions of closure and maintainability.

Definition 0.7 Let $A = (X, \Sigma, f, d)$ be a system and S be a set of states. By $Closure(S, A)$ we refer to the set $\bigcup_{x \in S} R(A, x)$. $\square$

Definition 0.8 Given a system $A = (X, \Sigma, f, d)$, a set of control actions $U \subseteq \Sigma$, a specification of exogenous actions e, and a set of states E, we say a set of states S is *k-maintainable* with respect to E if there exists a feedback control K such that from each state x in $Closure(S, A_K)$, we will get to a state in E with at most k transitions, where each action (behind the transitions) is dictated by the control K.

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 $S = X$, then we say A is maintainable with respect to E. $\square$

We now show that while stabilizability guarantees maintainability, the opposite is not true.

Proposition 0.1 Given a system A, if a set of states S is stabilizable with respect to a set of states E, then S is maintainable with respect to E. $\square$

Proof : Suppose that a set $S \subseteq X$ and S is stabilizable with respect to E. Then there exists a control law K such that for each $x \in S$, x is alive and is stable with respect E.
Claim: There is a trajectory from each state x in S to a state in E with a finite transitions.
Case 1. Suppose $x \in S$. Then x is stable, therefore we can get from x to a state in E with a finite number of transitions dictated by K, say n_x transitions.
Case 2. Suppose $x \in Closure(S, A) \backslash S$. Then there exists $y \in S$ such that there is a trajectory T from y which goes through x. Since $y \in S$, y is stabilizable. Thus all trajectories consistent with A and starting from y go through a state in E in a finite number of transitions and they visit E infinitely often afterwards. Therefore any trajectory from y which goes through x will visit E infinitely. Thus there must be a sub trajectory T' from x to a state in E which is contained in the trajectory T from y to a state in E through x. Through this trajectory T', we can reach from x to a state in E in a finite number of transitions dictated by K, say n_x. Note that the maximum possible cardinality of $Closure(S, A)$ is the cardinality of X. Thus it is finite. Let n be $max\{n_x | x \in Closure(S, A)\}$. Since $Closure(S, A)$ is finite, n exists ($n < \infty$) and from all states in $Closure(S, A)$ we can reach a state in E within n transitions dictated by K. Hence S is n-maintainable with respect to E and thus S is maintainable with respect to E. $\square$

But the converse of the above proposition is not true. I.e. *Maintainability does not necessarily imply stabilizability.* We now show an example of a system which is maintainable but is not stabilizable.

Consider a system $A = (X, \sum, f, d)$ with the following:
$X = \{s_1, s_2, s_3, s_4, s_5\}$,
$\sum = \{a_1, a_2, a_3, a_4, a_5\} \bigcup \{e_1, e_2\}$,

$d(s_1) = \{a_1\}$, $d(s_2) = \{a_2, e_1, e_2\}$, $d(s_3) = \{a_3\}$, $d(s_4) = \{a_4\}$, $d(s_5) = \{a_5\}$

$f(s_1, a_1) = \{s_2\}$, $f(s_2, a_2) = \{s_4\}$, $f(s_2, e_1) = \{s_3\}$, $f(s_2, e_2) = \{s_2\}$, $f(s_3, a_3) = \{s_4\}$, $f(s_4, a_4) = \{s_5\}$, $f(s_5, a_5) = \{s_4\}$

Given $E = \{s_4, s_5\}$, this system is maintainable, but is not stabilizable. With the control law K, where $K(s_i) = a_i$, with at most 3 transitions, we can reach from any state in X to a state in E, therefore it is maintainable. But if we consider all trajectories, at the state s_2, the exogenous action e_2 can keep interfering and we might never reach from the state s_2 to a state in E. Therefore it is not stabilizable. $\square$

Maintainability in an active database

In this section we show how the notion of 'maintainability' is useful in defining the correctness of an active database.

Consider an active database with the following aspects:

- Relational Schema:

  $Employee(Emp\#, Name, Salary, Dept\#)$
  $Dept(Dept\#, Mgr\#)$

- Goal of the active database: Maintain Integrity constraints. I.e., Maintain the database in states where

  (i) If (e, n, s, d) is a tuple in $Employee$ then there must be a tuple (d', m') in $Dept$ such that $d = d'$; and

  (ii) If (d, m) is a tuple in $Dept$, then there must be a tuple (e', n', s', d') in $Employee$ such that $d = d'$ and $m = e'$

  (In addition we may have other constraints – which we do not focus here – such as each department has a single manager and each employee works in a single department.)

- Exogenous actions are of the kind: Delete (E, N, S, D) from $Employee$. (The direct effect of this action is the deletion of the tuple.)

- Triggers are of the kind:
  1. For any Delete (e, n, s, d) from $Employee$, if (d, e) is a tuple in $Dept$, delete that tuple from $Dept$ and delete all tuples of the form (e', n', s', d') from $Employee$, where $d = d'$.

To formulate the correctness of such an active database, we can treat the triggers as control laws, as was done initially in (Ceri & Widom 1990). The overall system operates in a way that whenever an exogenous action occurs if it modifies the database such that integrity constraints are violated, the triggers (control laws) kick in and force additional changes to the database such that it reaches a state where the integrity constraints are satisfied. This can be formulated as *maintenance of the integrity constraints.*

Now, if there were a continuous stream of exogenous actions (whose direct effects were immediately reflected in the database) then there is no guarantee that the database would reach a state satisfying the integrity constraints within a finite number of transitions. Hence, we can not formulate this as stabilizability.

Another important aspect of maintainability is that in reactive $software$ systems like this, if we know that our system is k-maintainable, and each transition takes say at most t time units, then we can implement a transaction mechanism that will regulate the number of exogenous actions allowed per unit time to be $\frac{1}{k \times t}$. This will also be useful in web-based transaction softwares where exogenous actions are external interactions and the internal service mechanism is modeled as control laws. On the other hand, given a requirement that we must allow m requests (exogenous actions) per unit time, we can work backwards to determine the value of k, and then find a control to make the system k-maintainable. In general, since in high level controls we may have the opportunity to limit (say through a transaction mechanism) the exogenous actions, we think 'maintainability' is an important notion for high level control.

Algorithms

In this section we give two simple algorithms to verify maintainability, and to generate control for maintainability. We will further analyze them in the full paper.

Testing maintainability

Input: A system $A = (X, \Sigma, f, d)$, a set of states E, a set of states S, and a control K.

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

Algorithm:

Step 1: Compute $Closure(S, A_K)$.

Step 2: For each x in $Closure(S, A_K)$ compute the sequence
$x_0, x_1, \ldots, x_k, x_{k+1}, \ldots, x_{|X|}$, where $x_0 = x$, and $x_{k+1} = x_k$ if $x_k \in E$, and $x_{k+1} = f(x_k, K(x_k))$ otherwise.

Step 3: If for all x, $\{x_0, \ldots, x_{|X|}\} \cap E \neq \emptyset$ 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.

Generating control for maintainability of a set of states

Input: A system $A = (X, \Sigma, f, d)$, a set of states E, and a set of states S.

Output: Find 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.

Pick an x from $S_{in} \setminus S_{out}$. Find a *shortest path (or a minimal cost path)* from x to a state in E using only control actions.

If no such path exists then EXIT and return(FAIL).

Let a be the first action of that path.
Assign $K(x) = a$.
$S_{out} := S_{out} \cup \{x\}$
$S_{in} := S_{in} \cup \{f(x, a)\} \cup \{x : x \in f(x, b)$, for some $b \in e(X)\}$.

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

Proposition 0.2 If the above algorithm terminates by returning S' and K, then: (i) $S' = Closure(S, A_K)$, and (ii) S is maintainable with respect to E, using the control K. $\square$

One important aspect of the above algorithm and its proof of correctness is the requirement of picking the first action of a shortest path or a minimal cost path. Picking the first action of a minimal path (as normally used in the notions of minimal plans) will not be sufficient as that may lead to cycles and the system may never reach its goal. An algorithm based on a minimal path will have to be more complicated so as to avoid this. On the other hand, our use of shortest path allows us to easily enhance the control when additional states are added to S. We then only need to consider the new states in the closure, find shortest paths from each of these states (say x), and have the first action as the value of $K(x)$. Thus our algorithm is useful in incrementally broadening the control when the set of initial states S is broadened.

At this point we would like to point out the relation between our work here and some research on reactive and situated agents (Kaelbling & Rosenschein 1991). In (Kaelbling & Rosenschein 1991), 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 cost plan.

Supportability: a notion that generalizes stabilizability and 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. We refer to this general notion as *supportability*.

Definition 0.9 Given a system $A = (X, \Sigma, f, d)$, a set of agents action $U \subseteq \Sigma$, a specification of exogenous actions e, and a set of states E, we say a set of states S is *(k,l)-supportable* ($l \leq k$) with respect to E if there exists a control law K such that for each state x in $Closure(S, A_K)$, all trajectories – consistent with A_K – from x 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 0.3 $(k, 0)$-supportable is equivalent to k-maintainable. (A set of states S is $(k, 0)$-supportable with respect to a set of states E if and only if S is k-maintainable with respect to E.)

Proposition 0.4 A set of states S is stabilizable iff S is alive and there exists an integer m such that S is *(m,m)-supportable* with respect to E.

An automata and a temporal logic for 'maintainability' and 'supportability'

The notion of a system defined earlier does not distinguish between exogenous action and control action. They are both part of Σ. In this section we first define the notion of a 2-system where we distinguish between exogenous and control actions. Using the notion of a two system we define the notion of 'maintained' which is analogous to the notion of being 'stable' and related it to our earlier notion of maintainability. We then use the notion of 2-systems to define a temporal logic that makes the distinction between transitions due to exogenous action and transitions due to control actions.

Definition 0.10
A 2-system A is a 5-tuple $(X, \Sigma_a, \Sigma_e, f, d)$, where X is a finite set of states, Σ_a is a finite set of control actions, Σ_e is a finite set of control events, d is a function from X to $2^{\Sigma_a \cup \Sigma_e}$ listing what actions and events may occur (or are executable) in what state, and f is a transition function from X and $\Sigma_a \cup \Sigma_e$ to 2^X. □

The notion of a trajectory with respect to a 2-system remains the same as with respect to a system, which we earlier defined in Definition 0.2.

Definition 0.11 Given a 2-system A and a set of states E, a state x is said to be *k-maintained* in A w.r.t. E if for all trajectories of the form $x = x_0, a_1, x_1, a_2, \ldots, a_j, x_j, a_{j+1}, \ldots$ that is consistent with A and for all i such that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \Sigma_a$, we have that $\{x_{i+1}, \ldots, x_{i+k}\} \cap E \neq \emptyset$.

A 2-system $A = (X, \Sigma_a, \Sigma_e, f, d)$ is k-maintained with respect to E if all its states are k-maintained.

A 2-system $A = (X, \Sigma_a, \Sigma_e, f, d)$ is maintained with respect to E if there exists a positive integer n such that it is n-maintained with respect to E. □

Proposition 0.5 A state x is k-maintainable in a system $A = (X, \Sigma_a \cup \Sigma_e, f, d)$ with respect to E iff there exists a control law K such that x is k-maintained with respect to E in the 2-system $A_K = (X, \Sigma_a, \Sigma_e, f, d_K)$, where d_K is as defined earlier in Definition 0.5.

A temporal language with respect to 2-systems

In the past, temporal logic has been used to specify and verify the behavior of reactive systems (Manna & Pnueli 1992; Clarke, Emerson, & Sistla 1986; Kabanza, Barbeau, & St-Denis 1997). Most of these temporal logics do not (perhaps with the exception of one description in (Singh 1994)) distinguish between transitions due to control actions and due to exogenous actions. Hence, they are too strong to be able to characterize the correctness of reactive software systems such as an active database system. In this section we propose a temporal language that makes a distinction between transitions due to control actions and exogenous actions and is able to characterize correctness of reactive software systems such as an active database system. We plan to elaborate on this in the full paper.

Some of the important future temporal operators as discussed in (Manna & Pnueli 1992) are: Next ($\bigcirc$), Always ($\Box$), Eventually ($\diamond$), and Until ($\mathcal{U}$). There meaning with respect a trajectory $\sigma = x_0, a_1, x_1, \ldots, x_j, a_{j+1}, x_{j+1}, \ldots$ is defined as follows:

- $(\sigma, j) \models p$ iff p is true in x_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, $j \leq i < k$, $(\sigma, i) \models p$.

It is easy to see that none of the above temporal operators consider the action type (whether exogenous or control action) behind the transitions. We now introduce some temporal operators that do consider the action type behind the transitions.

- $(\sigma, j) \models \bigcirc_k p$ iff $i \geq j$ is the smallest index such that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \Sigma_a$ and $(\sigma, i+r) \models p$, for some $1 \leq r \leq k$.
- $(\sigma, j) \models \Box_k p$ iff for all $i \geq j$ if $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \Sigma_a$ then $(\sigma, i+r) \models p$ for some $1 \leq r \leq k$.
- $(\sigma, j) \models \bigcirc_{k,l} p$ iff $i \geq j$ is the smallest index such that $|\{a_{i+1}, \ldots, a_{i+k}\} \cap \Sigma_e| \leq l$ and $(\sigma, i+r) \models p$ for some $1 \leq r \leq k$.
- $(\sigma, j) \models \Box_{k,l} p$ iff for all $i \geq j$ if $|\{a_{i+1}, \ldots, a_{i+k}\} \cap \Sigma_e| \leq l$ then $(\sigma, i+r) \models p$ for some $1 \leq r \leq k$.

We can describe the intuitive meaning behind the above formal definitions as follows: Intuitively, $(\sigma, j) \models \Box_k p$ means that starting from x_j, within or after any k consecutive transitions due to control actions p holds. Similarly, $(\sigma, j) \models \Box_{k,l} p$ means that starting from x_j, within or after any k transitions with at most l exogenous actions p holds.

Proposition 0.6 (i) $(\sigma, j) \models \Box_k p$ iff $(\sigma, j) \models \Box_{k,0} p$.

(ii) $(\sigma, j) \models \bigcirc_k p$ iff $(\sigma, j) \models \bigcirc_{k,0} p$.

(iii) Let E_p be the set of states, where a formula p holds. S is (k,l)-supportable w.r.t. E_p iff for all trajectories σ whose $x_0 \in S$, and for all j, $(\sigma, j) \models \Box_{k,l} p$. $\Box$

Corollary 0.7 1. Let E_p be the set of states, where a formula p holds. S is k maintainable w.r.t. E_p iff for all trajectories σ whose $x_0 \in S$, and for all j, $(\sigma, j) \models \Box_k p$.

2. Let E_p be the set of states, where a formula p holds. S is stabilizable w.r.t. E_p iff S is alive and there exists an m such that for all trajectories σ whose $x_0 \in S$, and for all j, $(\sigma, j) \models \Box_{m,m} p$.

Conclusion and related work

In this paper we formalized the notion of 'maintenance' often mentioned (Baral & Son 1998) in the context of robots and agents, as a property of a discrete event dynamic system (DEDS) and compared it with the notion of 'stability' and 'stabilizability' that are most popular in DEDS. We argued why 'maintainability' may be a more preferred notion for certain systems and discussed active database systems as an example. We then gave simple algorithms for testing maintainability and generating control for maintainability. We then developed the notion of 'supportability' that generalizes both 'maintainability' and 'stabilizability. Finally, we developed an automata theory that distinguishes between exogenous and control actions, and developed a temporal logic based on it. Our basic formulation of 'maintainability' is related to the work in (Baral & Son 1998).

Among the other related works, there has been some work on defining stability of continuous systems in the presence of discontinuities and disturbances; for example (Sontag 1999). In the planning literature there has been some work on planning for temporal goals (Bacchus & Kabanza 1998; Weld & Etzioni 1994) where goals are expressed as temporal formulas. But they use the traditional temporal operators which by themselves can not express our notion of 'maintenance'. Another related notion is planning from the current situation in a dynamic domain (Baral, Gelfond, & Provetti 1997) and execution monitoring (DeGiacomo, Reiter, & Soutchanski 1998). In both these notions 'maintenance' is achieved by monitoring (or observing) the world for discrepancies and making new plans to recover. Finally, the notion of 'self-stabilization' (Dijkstra 1974) in distributed and fault-tolerant computing seems to be similar to our notion of 'maintenance' and we plan to compare and contrast them in the sequel.

References

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

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

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

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

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

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

DeGiacomo, G.; Reiter, R.; and Soutchanski, M. 1998. Execution monitoring of high-level robot programs. In *Proc. of KR 98*, 453–464.

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

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

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

Maes, P., ed. 1991. *Designing Autonomous Agents*. MIT/Elsevier.

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

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

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

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

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

Singh, M. 1994. *Multiagent systems - a theoretical framework for intentions, know-how, and communications*. Springer-Verlag.

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

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

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