Maintenance Goals of Agents in a Dynamic Environment: Formulation and Policy Construction*

Chitta Baral†, Thomas Eiter♯, Marcus Bjäreland‡, and Mutsumi Nakamura†

† Department of Computer Science and Engineering,
Arizona State University, Tempe, AZ 85233, USA.
{*chitta,mutsumi*}*@asu.edu*

‡ AstraZeneca R&D
S-43183 Mölndal, Sweden
marcus.bjareland@astrazeneca.com

♯ Institute of Information Systems,
Vienna University of Technology, A-1040 Vienna, Austria
eiter@kr.tuwien.ac.at

Keywords: intelligent agents, maintenance goals, maintainability, agent control, policy construction, declarative logic programming, SAT solving, computational complexity, discrete event dynamic systems.

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 are not intuitive to characterize the maintenance behavior of certain agents in a dynamic environment. We propose a different characterization of maintenance and distinguish it from earlier notions such as *stabilizability*. Our notion of maintenance is more sensitive to a good-natured agent which struggles with an "adversary" environment, which hinders her by unforeseeable events to reach her goals (not in principle, but in case). It has a parameter k, referring to the length of non-interference (from exogenous events) needed to maintain a goal; we refer to this notion as k*-maintainability*. We demonstrate the notion on examples, and address the important but non-trivial issue of efficient construction of maintainability control functions. We present an algorithm which in polynomial time constructs a k-maintainable control function, if one exists, or tells that no such control is possible. Our algorithm is based on SAT Solving, and employs a suitable formulation of the existence of k-maintainable control in a fragment of SAT which is tractable. For small k (bounded by a constant), our algorithm is linear time. We then give a logic programming implementation of our algorithm and use it to give a standard procedural algorithm, and analyze the complexity of constructing k-maintainable controls, under different assumptions such as $k = 1$, and states described by variables. On the one hand, our work provides new concepts and algorithms for maintenance in dynamic environment, and on the other hand, a very fruitful application of computational logic tools.

*A preliminary version of the formulation part, entitled "A formal characterization of maintenance goals," has been presented at AAAI'00, and a preliminary version of the algorithm part entitled "A polynomial time algorithm for constructing k-maintainable policies" has been presented at ICAPS'04. The current version revises and combines both of them with additional elaborations, examples, results, and proofs. The major part of the algorithms was done when Chitta Baral was visiting Vienna University of Technology during May 2003. Marcus Bjäreland carried out the major part of his work while he was with the Department of Computer and Information Science of Linkoping University.

1 Introduction and Motivation

For an agent situated in a static environment, the goal is often to reach one out of several states where certain conditions are satisfied. Such a goal is usually expressed by a formula in propositional or first-order logic. Sometimes the goal requires constraining the path taken to reach one of the states. In that case, the goal can be expressed by a formula in temporal logic [1, 41, 4].

Our concern in this paper is about agents in a dynamic environment. In that case, things are more complex since the state of the world can change through both actions of the agent and of the environment. The agent's goal in a dynamic environment is then often more than just achieving a desired state, as after the agent has successfully acted to reach a desired state, the environment may change that state. In such a case, a common goal of an agent 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*. Maintenance goals are well-known in the AI literature, e.g., [52, 30, 1, 42], and have counterparts in other areas such as in stability theory of discrete event dynamic systems [43, 45, 47, 46, 51] and in active databases [10, 38]. However, as we argue in this paper, earlier characterizations of maintenance goals are not adequate under all circumstances.

To see what is wrong with earlier definition of maintenance goals, suppose an agent's goal is to maintain a fluent f, i.e., the proposition f should be true. A straightforward attempt[1] 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 means that things go out of shape and they have to be maintained back to shape. A better temporal logic representation of this goal is thus 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 at any stage $i \geq 0$, there exists some stage $j \geq i$ 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', if we take on a more refined view of the (sometimes nasty) part which the environment might play, which we illustrate by some examples. 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 the agent gets, we would then like to say that agent maintains the Inbox empty. But such a control does not satisfy the formula $\Box\Diamond f$ under all circumstances, because there will be trajectories where the agent is overwhelmed by the environment (flooding the Inbox) and f never becomes $true$.

Another example in support of our intuition behind maintainability is the notion of maintaining the consistency of a database [10, 38, 53]. 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 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

[1] All through the paper we consider the evaluation of linear temporal formulas with respect to all 'valid' trajectories. An alternative approach would be to use a variation of the branching time quantifier A, such as the operator A_π from [6], before the linear temporal formulas.

window of opportunity or a respite from continuous harassment from the environment to bring about the additional changes which are necessary to restore database consistency. The same holds for maintaining a room clean; we can not fault the cleaning person if he or she is continually sent away because the room is being continuously used.

Another example is a mobile robot [8, 35] which is asked to 'maintain' a state where there are no obstacles in front of it. Here, if there is a belligerent adversary that keeps on putting an obstacle in front of the robot, there is no way for the robot to reach a state with no obstacle in front of it. But often we will be satisfied if the robot avoids obstacles 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 obstacles 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 actions and environment actions. Thus we can not distinguish the cases

(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) 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 notions of temporal logics [12, 36], 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.

The main contributions of this paper can be summarized as follows.

1. We introduce and formally define the notion of k-maintainability, and distinguish it from earlier notions of maintainability, in particular the specification $\Box\Diamond f$ and the similar notion of stabilizability from discrete event dynamic systems.

2. We provide polynomial time algorithms that can construct k-maintainable control policies, if one exists. (In the rest of the paper we will refer to 'control policy' simply by 'control'.) Our algorithm is based on SAT Solving, and employs a suitable formulation of the existence of k-maintainable control in a tractable fragment of SAT. We then give a logic programming implementation of this method, and finally distill from it a standard procedural algorithm.

3. We analyze the computational complexity of constructing k-maintainable controls, under different settings of the environment and the windows of opportunity open to the agent, as well as under different forms of representation. We show that the problem is complete for **PTIME** in the standard setting, where the possible states are enumerated, and complete for **EXPTIME** in a STRIPS-style setting where states are given by value assignments to fluents. Furthermore, we elucidate the impact of the different factors and show, by our proofs of the hardness results, that the full problem complexity is inherent already to certain restricted cases.

Overall, our work not only provides new concepts and algorithms for realizing maintenance of an agent in dynamic environment, but also illustrates a very fruitful application of computational logic tools.

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 describe a running example of a system with two buffers. We use this example for illustrating the concepts of stabilizability and k-maintainability, which is formally defined in Section 4. In Section 5 we present our algorithms for constructing k-maintaining controls, based on SAT Solving as well as a genuine algorithm extracted from it. In Section 6 we present an encoding for computing a control function using a logic programming engine and devote Section 7 to complexity analysis. Finally, in Section 8 we conclude, mention related work and outline some future directions.

2 Background: Systems, Goals, Control, Stability and Stabilizability

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 changes can happen to the state of the world beyond the control of an agent. The agent's job is thus to make the world evolve in a way coherent with a goal assigned to it. As for the agent control, we adopt here that an agent follows a Markovian control policy to do its job; that is, its control is a function from the set of states to the set of actions, detailed as follows.

Definition 1 (System) A *system* is a quadruple $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, where

- $\mathcal{S}$ is the set of system states;
- $\mathcal{A}$ is the set of actions, which is the union of the set of agents actions, $\mathcal{A}_{agent}$, and the set of environmental actions, $\mathcal{A}_{env}$;
- $\Phi : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ is a non-deterministic transition function that specifies how the state of the world changes in response to actions; and
- $poss : \mathcal{S} \rightarrow 2^{\mathcal{A}}$ is a function that describes which actions are possible to take in which states.

The above notion of system is used in the discrete event dynamic systems community, for instance in [43, 45, 47, 46, 51]. In practice, the functions Φ and $poss$ are required to be effectively (and efficiently) computable, and they may often be specified in a representation language such as in [25, 23, 48]. The possibility of an action has different meaning 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. We tacitly assume here that possible actions lead always to some successor state, i.e., the axiom that $\Phi(s, a) \neq \emptyset$ whenever $a \in poss(s)$ holds for any state s and action a, is satisfied by any system.

An example of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, where $\mathcal{S} = \{b, c, d, f, g, h\}$, $\mathcal{A} = \{$ **a**, **a**′, **e**$\}$, and the transition function Φ is shown in Figure 1, where $s' \in \Phi(s, a)$ iff an arc $s \rightarrow s'$ labeled with a is present and $poss(s)$ are all actions that label arcs leaving s. Notice that in this example, $\Phi(s, a)$ is *deterministic*, i.e., $\Phi(s, a)$ is a singleton if nonempty.

The evolution of the world with respect to a system is characterized by the following definition.

Definition 2 (Trajectory) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, an alternating infinite 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, if $s_{k+1} \in \Phi(s_k, a_{k+1})$, and $a_{k+1} \in poss(s_k)$. □

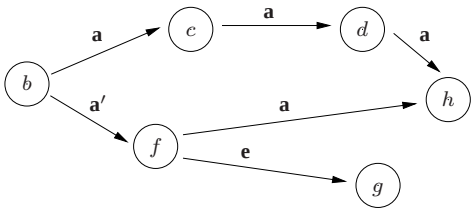

Figure 1: Transition diagram of system A

A common restriction on how the world evolves is defined using the notion of *stability*. The following definition of stability is adapted from [43] and has its origin in control theory and discrete event dynamic systems [43, 45, 47, 46].

Definition 3 (Stable state 1) 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. □

Although the above definition of stability is with respect to a set of states E, it can be easily adapted to a formula φ that can be evaluated at the states of system A. In that case $E = \{s \in \mathcal{S} \mid A, s \models \varphi\}$, i.e., it is the set of states s at which φ is true.

An alternative approach to characterize the evolution of states is through temporal operators. Some of the important temporal operators talking about the future are (cf. [36, 21]): Next ($\bigcirc$), Always ($\Box$), Eventually ($\Diamond$), and Until ($\mathcal{U}$). Their meaning with respect a trajectory $\tau = s_0, a_1, s_1, \ldots, s_k, a_{k+1}, s_{k+1}, \ldots$ is defined as follows.

Let (τ, j), for $j \geq 0$, denote the remainder of τ starting at s_j; then

- $(\tau, j) \models p$ iff p is true in s_j, for any proposition p;
- $(\tau, j) \models \bigcirc\phi$ iff $(\tau, j+1) \models \phi$;
- $(\tau, j) \models \Box\phi$ iff $(\tau, k) \models \phi$, for all $k \geq j$.
- $(\tau, j) \models \Diamond\phi$ iff $(\tau, k) \models \phi$, for some $k \geq j$.
- $(\tau, j) \models \phi_1 \,\mathcal{U}\, \phi_2$ iff there exists $k \geq j$ such that $(\tau, k) \models \phi_2$ and for all i, $j \leq i < k$, $(\tau, i) \models \phi_1$.

The standard Boolean connectives $\wedge$, $\vee$, and $\neg$ are defined as usual. An alternative definition of stability can then be given as follows:

Definition 4 (Stable state 2) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and an objective formula φ (i.e., without temporal operators), let $E_\phi = \{s \in \mathcal{S} \mid \phi \text{ is true in } s\}$. A state s is then said to be *stable* in A w.r.t. E if for all trajectories τ of the form $\tau = s_0, a_1, s_1, \ldots, s_k, a_{k+1}, s_{k+1}, \ldots$ consistent with A, it holds that $(\tau, 0) \models \Box\Diamond\varphi$. □

In fact, this definition is equivalent to Definition 3. The advantage of using temporal operators, as in the above definition, instead of Definition 3 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* [36], 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 an 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 $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, when discussing stabilizability of the system, we need to consider the following additional aspects:

- the set of actions $\mathcal{A}_{agent}$ which the agent is capable of executing in principle (where $\mathcal{A}_{agent} \subseteq \mathcal{A}$);
- the set of *exogenous actions* that may occur in the state s, beyond the agent's control, modeled by a function $exo : \mathcal{S} \rightarrow 2^{\mathcal{A}_{env}}$, where $exo(s) \subseteq poss(s)$ for each state s (recall that $\mathcal{A}_{env}$ are the environmental actions). We call any such exo an *exogenous function*.

Intuitively, given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, $\mathcal{A}_{agent}$, exo, and E, a state s is stabilizable with respect to E, if we are able to find a policy or *control function* such that the agent picks an action it can do in s, we have stability if all other agent actions in s and the other states that are reached are disabled, and no state is reached from s where no further actions are possible.

The last condition is referred to as aliveness. 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 5 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a state s, $R(A, s) \subseteq \mathcal{S}$ is the smallest set of states that satisfying the following conditions:

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

Definition 6 (Aliveness) 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. □

The notion of control function is formally defined as follows.

Definition 7 (Control) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions, a *control function for A w.r.t. $\mathcal{A}_{agent}$* is a partial function

$$K : \mathcal{S} \rightarrow \mathcal{A}_{agent},$$

such that $K(s) \in poss(s)$ whenever $K(s)$ is defined. □

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

Definition 8 (Stabilizability) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a function exo as above, 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 $K : \mathcal{S} \to \mathcal{A}_{agent}$ for A w.r.t. $\mathcal{A}_{agent}$ with the following properties:

1. s is stable with respect to E in the system $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$, where, for any state s', $poss_{K,exo}(s') = \{K(s')\} \cup exo(s')$; and
2. s is alive in $A_{K,exo}$.

A set of states $S \subseteq \mathcal{S}$ is *stabilizable with respect to* E, if there is a control function K for $\mathcal{A}$ w.r.t. $\mathcal{A}_{agent}$ such that every state $s \in S$ is stabilizable with respect to E witnessed by K. □

Having provided this definition, we shall illustrate it on an elaborated example in the next section, where we describe an intuitive control function for the management of two finite buffers.

Before closing this section, we introduce for later use the notion of a super control.

Definition 9 (Super-control) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions, a partial function $K : \mathcal{S} \to 2^{\mathcal{A}_{agent}}$ such that $K(s) \subseteq poss(s)$ and $K(s) \neq \emptyset$ whenever $K(s)$ is defined, is called *super-control* for A w.r.t. $\mathcal{A}_{agent}$. □

Informally, a super-control is an envelope for multiple control functions, which result by refining K to some arbitrary action in $K(s)$ whenever $K(s)$ is defined; the notion of stabilizability is defined similar as for control functions, with the only change that in $A_{K,exo}$, we set $poss_{K,exo}(s') = K(s') \cup exo(s')$ in place of $poss_{K,exo}(s') = \{K(s')\} \cup exo(s')$.

The following proposition is immediate.

Proposition 1 *Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a function exo, a set of states $S \subseteq \mathcal{S}$ is stabilizable with respect to a set of states $E \subseteq \mathcal{S}$ under a control function K for A w.r.t. $\mathcal{A}_{agent}$ iff S is stabilizable with respect to E under a super-control K^+ for A w.r.t. $\mathcal{A}_{agent}$. Furthermore, each such K is a refinement of some K^+ with this property (i.e., for each s, $K(s) \in K^+(s)$ and $K(s)$ is defined iff $K^+(s)$ is defined), and each refinement K of K^+ is a control function witnessing stabilizability of S with respect to E.*

3 Example Scenario: 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 in 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 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 [45] 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.

In our framework, we shall describe a system A_b which models this scenario. 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 $\mathbf{Proc}$ 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.

Let us assume that the control goal of this system is to keep b_1 empty. Then, 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 a concrete instance of A_b.

Example 1 (Running Example)
We assume that the maximum capacity of the buffers b_1 and b_2 is 3. The components of $A_b = (\mathcal{S}_b, \mathcal{A}_b, \Phi_b, poss_b)$ are then as follows.

- We model every state by the current number of objects in b_1 and b_2. That is, a state s is identified by a pair of integers $\langle i, j\rangle$ where i denotes the number of objects in b_1 and j the number of objects in b_2. With the maximum capacity of 3, the set of states, $\mathcal{S}_b$, consists of $4 \times 4 = 16$ states and is given by
$$\mathcal{S}_b = \{0,1,2,3\} \times \{0,1,2,3\}.$$
- The set of actions is $\mathcal{A}_b = \{\mathbf{M}_{12}, \mathbf{M}_{21}, \mathbf{Proc}, Ins\}$.
- We assume that the transition function Φ_b is deterministic, i.e., $|\Phi_b(s,a)| \leq 1$, defined as follows, where we write $\Phi_b(s,a) = s'$ for $\Phi_b(s,a) = \{s'\}$. For every $i, j \in \{0,\ldots,3\}$, let
$$\begin{aligned}
&\Phi_b(\langle i,j\rangle, \mathbf{M}_{12}) = \langle i-1, j+1\rangle\\
&\Phi_b(\langle i,j\rangle, \mathbf{M}_{21}) = \langle i+1, j-1\rangle,\\
&\Phi_b(\langle i,j\rangle, \mathbf{Proc}) = \langle i, j-1\rangle,\\
&\Phi_b(\langle i,j\rangle, Ins) = \langle i+1, j\rangle,
\end{aligned}$$
where addition and subtraction are modulo 3, and and in all other cases $\Phi_b(s,a) = \emptyset$.
- The enabling function, $poss_b$, is defined by
$$\begin{aligned}
&\mathbf{M}_{12} \in poss_b(\langle i,j\rangle) \quad \text{iff} \quad i \geq 1 \text{ and } j \leq 2\\
&\mathbf{M}_{21} \in poss_b(\langle i,j\rangle) \quad \text{iff} \quad i \leq 2 \text{ and } j \geq 1\\
&\mathbf{Proc} \in poss_b(\langle i,j\rangle) \quad \text{iff} \quad j \geq 1\\
&Ins \in poss_b(\langle i,j\rangle) \quad \text{iff} \quad i \leq 2
\end{aligned}$$

It is easy to see that for $S = \{\langle 0,0\rangle\}$ (no objects in the buffers) and $E = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle\}$ (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:

$$\begin{aligned}
&K_b(\langle i,j\rangle) = \mathbf{M}_{12} \text{ when } i \geq 1 \text{ and } j < 3, \text{ and}\\
&K_b(\langle i,j\rangle) = \mathbf{Proc} \text{ when } (i = 0 \text{ and } j \geq 1) \text{ or } j = 3.
\end{aligned}$$

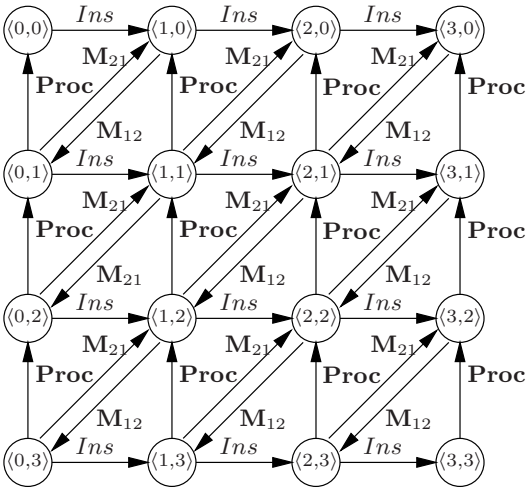

Figure 2: The transition diagram of the buffer system A_b for the concrete instance (buffer capacity 3).

Intuitively, the above control directs the transfer of objects from buffer 1 to 2 whenever possible, and if that is not possible it directs processing of objects in buffer 2 if that is possible. In Figure 1, which shows the transition diagram between states, the transitions by the control K_b are marked with $\mathbf{M}_{12}$ and $\mathbf{Proc}$.

Consider the following trajectory consistent with the control system $A_{K,exo} = (\mathcal{S}_b, \mathcal{A}_b, \Phi_b, poss_{b_{K_b,exo}})$:

$$\tau = \langle 0,0\rangle, Ins, \langle 1,0\rangle, Ins, \langle 2,0\rangle, \mathbf{M}_{12}, \langle 1,1\rangle, Ins, \langle 2,1\rangle, \mathbf{M}_{12}, \overline{\langle 1,2\rangle, Ins, \langle 2,2\rangle, \mathbf{M}_{12}, \langle 1,3\rangle, \mathbf{Proc}}.$$

It consists of a prefix $\langle 0,0\rangle, Ins, \ldots, \mathbf{M}_{12}$ and a cycle $\overline{\langle 1,2\rangle, \ldots, \mathbf{Proc}}$. In τ, no state in E is ever reached after the starting state $\langle 0,0\rangle$. Similar trajectories can be found for any control and hence S is not stabilizable with respect to E.

On the other hand, $S = \{\langle 0,0\rangle\}$ is stabilizable w.r.t. $E' = \{0,1,2\} \times \{0,1,2,3\}$ (that is, we want to have at most two objects in b_1 at any time): Following K_b we can go from any of the states in $\mathcal{S}_b \setminus E' = \{\langle 3,0\rangle, \langle 3,1\rangle, \langle 3,2\rangle, \langle 3,3\rangle\}$ to E' with the execution of at most two control actions, while no exogenous actions are possible for those states. □

4 Limited Interference and 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 maintenance is performed by the agent. 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 within those k steps.

Our definition of maintainability has the following parameters:

(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 $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions,

(v) a function $exo : \mathcal{S} \rightarrow 2^{\mathcal{A}_{env}}$ detailing exogenous actions, such that $exo(s) \subseteq poss(s)$, and

(vi) a control function K (mapping a relevant part of $\mathcal{S}$ to $\mathcal{A}_{agent}$) such that $K(s) \in poss(s)$.

The next step is to formulate when the control K maintains E assuming that the system is initially in one of the states in S. The exogenous actions are accounted for by defining the notion of a closure of S with respect to the system $A_{K,exo} = (\mathcal{S}, \mathcal{A}, \Phi, poss_{K,exo})$, denoted by $Closure(S, A_{K,exo})$; where $poss_{K,exo}(s)$ is the set $\{K(s)\} \cup exo(s)$. This closure is the set of states that the system may get into starting from S because of K and/or exo. Maintainability is then defined by requiring the control to 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 during that window. *One of the importance of using the notion of closure is that one can focus only on a possibly smaller state of states, rather than all the states, thus limiting the possibility of an exponential blow-up - as warned in [26] - of the number of control rules.*

Now a next question might be: Suppose the above condition of maintainability is satisfied, and while the control 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 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 10 (Closure) Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system and let $S \subseteq \mathcal{S}$ be a set of states. Then the closure of A w.r.t. S, denoted by $Closure(S, A)$, is defined by $Closure(S, A) = \bigcup_{s \in S} R(A, s)$. □

Example 2 In the system A in Figure 1, we have that $R(A, d) = \{d, h\}$ and $R(A, f) = \{f, g, h\}$, and therefore $Closure(\{d, f\}, A) = \{d, f, g, h\}$. □

We note some properties of $Closure(S, A)$, which follow immediately from the definition of $R(A, s)$.

Lemma 2 Let $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ be a system and $S \subseteq \mathcal{S}$ be a set of states. Then,

1. $Closure(S, A)$ satisfies the Kuratowski closure axioms [32], i.e., $Closure(\emptyset, A) = \emptyset$, $S \subseteq Closure(S, A)$, $Closure(Closure(S, A), A) = Closure(S, A)$, and $Closure(S_1 \cup S_2, A) = Closure(S_1, A) \cup Closure(S_2, A)$);
2. if $s \in Closure(S, A)$, and $a \in poss(s)$, then $\Phi(s, a) \subseteq Closure(S, A)$. □

Next we define the notion of unfolding a control.

Definition 11 ($Unfold_k(s, A, K)$**)** Let A=($\mathcal{S}$, $\mathcal{A}$, Φ, $poss$) be a system, let $s \in \mathcal{S}$, and let K be a control for A. Then $Unfold_k(s, A, K)$ is the set of all sequences $\sigma = s_0, s_1, \ldots, s_l$ where $l \leq k$ and $s_0 = s$ such that $K(s_j)$ is defined for all $j < l$, $s_{j+1} \in \Phi(s_j, K(s_j))$, and if $l < k$, then $K(s_l)$ is undefined. □

Intuitively, an element of $Unfold_k(s, A, K)$ is a sequence of states of length at most $k+1$ that the system may go through if it follows the control K starting from the state s. The above definition of $Unfold_k(s, A, K)$ is easily extended to the case when K is a super-control, meaning $K(s)$ is a set of actions instead of a single action. In that case, we overload Φ and for any set of actions a^*, define $\Phi(s, a^*) = \bigcup_{a \in a^*} \Phi(s, a)$.

We now define the notion of k-maintainability. This definition can be used to verify the correctness of a control.

Definition 12 (k-Maintainability) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a specification of exogenous action occurrence exo, we say that a control[2] K for A w.r.t. $\mathcal{A}_{agent}$ *k-maintains $S \subseteq \mathcal{S}$ with respect to $E \subseteq \mathcal{S}$*, where $k \geq 0$, if for each state $s \in Closure(S, A_{K,exo})$ and each sequence $\sigma = s_0, s_1, \ldots, s_l$ in $Unfold_k(s, A, K)$ with $s_0 = s$, it holds that $\{s_0, \ldots, s_l\} \cap E \neq \emptyset$.

We say that a set of states $S \subseteq \mathcal{S}$ (resp. A, if $S = \mathcal{S}$) is *k-maintainable*, $k \geq 0$, with respect to a set of states $E \subseteq \mathcal{S}$, if there exists a control K which k-maintains S w.r.t. E. K is then referred to as the witnessing control function. Furthermore, S (resp. A) is called *maintainable* w.r.t E, if S (resp. A) is *k-maintainable* w.r.t. E for some $k \geq 0$. □

We often will omit explicit mention of $\mathcal{A}_{agent}$, S, and E for control functions and maintainability if they are clear from the context.

Intuitively, the condition $\{s_0, s_1, \ldots, s_l\} \cap E \neq \emptyset$ above means that we can get from a state s_0 outside E to a state in E within at most k transitions—where each transition is dictated by the control K—if the world were to unfold as in $s = s_0, s_1, \ldots, s_l$. In particular, 0-maintainability means that the agent has nothing to do: after any exogenous action happening, the system will be in a state from E. Therefore, a trivial control K will do which is undefined on every state.

Example 3 Reconsider the system A in Figure 1. Let us assume that $\mathcal{A}_{agent}$ = { **a**, **a**′ }, that $exo(s)$ = { **e** } iff $s = f$ and that $exo(s)$ = $\emptyset$ otherwise. Suppose now that we want a 3-maintainable control policy for $S = \{b\}$ w.r.t. $E = \{h\}$. Clearly, such a control policy K is to take **a** in b, c, and d. Indeed, $Closure(\{b\}, A_{K,exo}) = \{b, c, d, h\}$ and $Unfold_3(b, A, K)$ = $\{\langle b, c, d, h\rangle\}$, $Unfold_3(c, A, K)$ = $\{\langle c, d, h\rangle\}$, and $Unfold_3(d, A, K)$ = $\{\langle d, h\rangle\}$; furthermore, each sequence contains h.

Suppose now that $\Phi(c, \mathbf{a})$=$\{d, f\}$ instead of $\{d\}$ (i.e., nondeterminism in c). Then, no k-maintainable control policy for $S = \{b\}$ w.r.t. $E = \{h\}$ exists for any $k \geq 0$. Indeed, the agent can always end up in the dead-end g. If, however, in addition $\Phi(g, \mathbf{a}') = \{f, h\}$ and $\mathbf{a}' \in poss(g)$, a 3-maintainable control policy K is $K(s)$ = **a** for $s \in \{b, c, d, f\}$ and $K(g)$= **a**′. □

Example 4 Buffer Example (cont'd)
Earlier we showed that in A_b, $S = \{\langle 0, 0\rangle\}$ is not stabilizable w.r.t. $E = \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 0, 2\rangle, \langle 0, 3\rangle\}$. Thus, we might ask whether S is at least maintainable w.r.t. E? The answer is positive: For the worst case system state, $\langle 3, 3\rangle$, a control can move the system to $\langle 3, 0\rangle$ (by three transitions executing $\mathbf{Proc}$) without interfering occurrences of exogenous actions. If there then are three further transitions without interference, the control can apply $\mathbf{M}_{12}$ three times and effect the state $\langle 0, 3\rangle$. This implies that S is 6-maintainable w.r.t. E. We can, with a similar argument show that A is 9-maintainable w.r.t. $\{\langle 0, 0\rangle\}$. A similar argument can be made with respect to the control K_b of Example 1.

[2]Note that here only $K(s)$ for $s \in Closure(S, A_{K,exo})$ is of relevance. For all other s, $K(s)$ can be arbitrary or undefined.

However, we have that A is *not* maintainable w.r.t., for example, $\{\langle 0,3\rangle\}$ (Since we cannot go from, for example, $\{\langle 0,0\rangle\}$, to $\{\langle 0,3\rangle\}$ with control actions only). □

As the above example points out, it is possible that S is maintainable but not stabilizable with respect to E. 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 every path between a state in S and a state in E involves 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. However, often for each exogenous action there are equivalent (in terms of effects) agent actions. In that case, any stabilizable system is also maintainable.

We note the following monotonicity property of k-maintainability, which is an easy consequence of the definition:

Proposition 3 Suppose that for a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a specification of exogenous action occurrence exo, the control function K k-maintains $S \subseteq \mathcal{S}$ w.r.t. $E \subseteq \mathcal{S}$. Then, K also k-maintains any set $S' \subseteq Closure(S, A_{K,exo})$ with respect to any set $E' \subseteq Closure(S, A_{K,exo})$ such that $E \subseteq E'$. □

4.1 An alternative characterization of k-maintainability

The characterization of stability and stabilizability in Section 2 is based on imposing 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.

In contrast, the definition of maintainability in Definition 12 is not based on trajectories. Nonetheless, one can give an alternative characterization based on trajectories, which we do next. To bridge from finite trajectories (which are relevant with respect to maintainability), to infinite ones as in Definition 2, we consider for each system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ an extension, A^{∞}, which results by adding a fresh environmental action a_{nop} such that in A^{∞}, for each state s we have $\Phi(s, a_{nop}) = \{s\}$ and $a_{nop} \in poss(s)$ if $poss(s) = \emptyset$ in A. Informally, A_{∞} adds infinite loops to halting states of A.

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 occurrence exo, and a set of states E, a set of states S is ***k-maintainable*** with respect to E, $k \geq 0$, if and only if there exists a control K for A w.r.t. $\mathcal{A}_{agent}$ such that for each state s in S and every trajectory of form $\tau = s_0, a_1, s_1, a_2, \ldots, a_j, s_j, a_{j+1}, \ldots$ consistent with $A^{\infty}_{K,exo}$ and $s_0 = s$, it holds that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$ or $a_{i+k} = a_{nop}$ for some $i \geq 0$ implies that $\{s_i, \ldots, s_{i+k}\} \cap E \neq \emptyset$. □

Proof. For the only if direction, suppose that S is k-maintainable w.r.t. E, witnessed by the control function K. Let $s \in S$ and $\tau = s_0, a_1, s_1, a_2, \ldots, a_j, s_j, a_{j+1}, \ldots$ be consistent with $A^{\infty}_{K,exo}$ such that $s_0 = s$ and $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$ or $a_{i+k} = a_{nop}$, for some $i \geq 0$. Then, we have $s_i \in Closure(S, A^{\infty}_{K,exo})$. If $k = 0$, then since K is a witnessing control, we have $s_i \in E$, and thus $\{s_i, s_{i+1}, \ldots, s_{i+k}\} \cap E \neq \emptyset$ holds. Consider thus $k > 0$. If $a_{i+k} \in \mathcal{A}_{agent}$ (which implies $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$), then the sequence $s_i, s_{i+1}, \ldots, s_{i+k}$ belongs to $Unfold_k(s_i, A, K)$. Since K is a witnessing control function, we again have $\{s_i, s_{i+1}, \ldots, s_{i+k}\} \cap E \neq \emptyset$. Otherwise, if $a_{i+k} = a_{nop}$, let $l \geq 1$ be the least index such that $a_l = a_{nop}$. By definition of $A^{\infty}_{K,exo}$, we have that $K(s_{l-1})$ is undefined. Hence, the sequence $\sigma = s_{l-1}$ belongs to $Unfold_k(s_{l-1}, A, K)$. Since K is a

control, it follows that $s_{l-1} \in E$. Since $s_j = s_{l-1}$ for each $j \geq l$, and in particular $s_{i+k} = s_{l-1}$, it follows again that $\{s_i, s_{i+1}, \ldots, s_{i+k}\} \cap E \neq \emptyset$. This proves the only if direction.

Conversely, suppose K is a control for A w.r.t. $\mathcal{A}_{agent}$ such that for each $s \in S$ and trajectory $\tau = s_0, a_1, s_1, a_2, \ldots, a_j, s_j, a_{j+1}, \ldots$ consistent with $A^{\infty}_{K,exo}$ and $s_0 = s$, it holds that $\{a_{i+1}, \ldots, a_{i+k}\} \subseteq \mathcal{A}_{agent}$ or $a_{i+k} = a_{nop}$ for some $i \geq 0$ implies that $\{s_i, s_{i+1}, \ldots, s_{i+k}\} \cap E \neq \emptyset$. We claim that K witnesses k-maintainability of S w.r.t. E. Towards a contradiction, suppose the contrary. Hence, it follows from the definition of $A^{\infty}_{K,exo}$, that there is some state $s \in S$ and trajectory $\tau = s_0, a_1, s_1, a_2, \ldots, a_j, s_j, a_{j+1}, \ldots$ consistent with $A^{\infty}_{K,exo}$ and $s_0 = s$, such that for some $j \geq 0$ we have $s_j \in \mathit{Closure}(S, A^{\infty}_{K,exo})$ and $s_j, s_{j+1}, \ldots, s_{j+l}$ is in $\mathit{Unfold}_k(s_j, A, K)$, where $l \leq k$, but $E \cap \{s_j, \ldots, s_{j+l}\} = \emptyset$.

By definition of $\mathit{Unfold}_k(s_j, A, K)$, we have that $\{a_{j+1}, \ldots, a_{j+l-1}\} \subseteq \mathcal{A}_{agent}$ and that $a_{j+l} = a_{j+l+1} = \cdots = a_{j+k} = a_{nop}$. By hypothesis, $E \cap \{s_j, \ldots, s_{j+k}\} \neq \emptyset$ holds. Thus, we conclude that $E \cap \{s_{j+l+1}, \ldots, s_{j+k}\} \neq \emptyset$ must hold, and hence $l < k$. However, by definition of $\Phi(s, a_{nop})$ we have $s_{j+l} = s_{j+l+1} = \cdots = s_{j+k}$. This implies that $E \cap \{s_j, \ldots, s_{j+l}\} \neq \emptyset$, which is a contradiction. This proves that K witnesses k-maintainability of S w.r.t. E. □

While this result shows that we could equally well have developed our notion of k-maintainability on the basis of trajectories, in the rest of this paper we shall stick to the setting which uses closure and unfolding. We find the latter more intuitive, as well as more convenient for designing algorithms and for proofs. Furthermore, this setting requires no special handling of possible finite trajectories, which complicates matters as becomes apparent from Proposition 4.

5 Polynomial Time Methods to Construct k-Maintainable Controls

Now that we have defined the notion of k-maintainability, our next step is to show how some k-maintainable control can be constructed in an automated way. We start with some historical background. There has been extensive use of situation control rules [17] and reactive control in the literature. But there have been far fewer efforts [30] to define correctness of such control rules[3], and to automatically construct correct control rules. In [31], it is suggested that in a control rule of the form: "if condition c is satisfied then do action a", the action a is the action that *leads to* the goal from any state where the condition c is satisfied. In [5] a formal meaning of "leads to" is given as: for all states s that satisfy c, a is the first action of a minimal cost plan from s to the goal. Using this definition, an algorithm is presented in [39] to construct k-maintainable controls. This algorithm is sound but not complete, in the sense that it generates correct controls only, but there is no guarantee that it will find always a control if one exists. The difficulty in developing a complete algorithm – also recognized in [29] in a slightly different context – can be explained as follows. Suppose one were to do forward search from a state in S. Now suppose there are multiple actions from this state that 'lead' to E. Deciding on which of the actions or which subsets one needs to chose is a nondeterministic choice necessitating backtracking if one were to discover that a particular choice leads to a state (due to exogenous actions) from where E can not be reached. Same happens in backward search too. *In this paper we overcome the problems one faces in following the straightforward approaches and give a sound and complete algorithm for constructing k-maintainable control policies.*

We provide it in two sets: First we consider the case when the transition function Φ is deterministic, and

[3]Here we exclude the works related to MDPs as it is not known how to express the kind of goal we are interested in – such as k maintenance goals – using reward functions.

then we generalize to the case where Φ may be non-deterministic. In each case, we present different methods, which illustrates our discovery process and also gives a better grasp of the final algorithm. We first present an encoding of our problem as a propositional theory and appeal to propositional SAT solvers to construct the control. As it turns out, this encoding is in a tractable fragment of SAT, for which specialized solvers (in particular, Horn SAT solvers) can be used easily. Finally, we present a direct algorithm distilled from the previous methods.

The reasoning behind this line of presentation is the following:

(i) It illustrates the methodology of using SAT and Horn SAT encodings to solve problems;

(ii) the encodings allow us to quickly implement and test algorithms;

(iii) the proof of correctness mimics the encodings; and

(iv) we can exploit known complexity results for Horn SAT to determine the complexity of our algorithm, and in particularly to establish tractability.

As for (ii), we can make use of Answer Set Solvers such as DLV [20, 33] or Smodels [40, 50] which extend Horn logic programs by nonmonotonic negation. These solvers allow efficient computation of the least model and some maximal models of a Horn theory, and can be exploited to construct robust or "small" controls, respectively.

The problem we want to solve, which we refer to as k-MAINTAIN, has the following input and output:

Input: An input I is a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, sets of states $E \subseteq \mathcal{S}$ and $S \subseteq \mathcal{S}$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, a function exo, and an integer $k \geq 0$.

Output: A control K such that S is k-maintainable with respect to E (using the control K), if such a control exists. Otherwise the output is the answer that no such control exists.

We assume here that the functions $poss(s)$ and $exo(s)$ can be efficiently evaluated; e.g., when both functions are given by their graphs (i.e., in a table).

5.1 Deterministic transition function $\Phi(s, a)$

We start with the case of deterministic transitions, i.e., $\Phi(s, a)$ is a singleton set $\{s'\}$ whenever nonempty. In abuse of notation, we simply will write $\Phi(s, a) = s'$ in this case.

Our first algorithm to solve k-MAINTAIN will be based on a reduction to propositional SAT solving. Given an input I for k-MAINTAIN, we construct a SAT instance $sat(I)$ in polynomial time such that $sat(I)$ is satisfiable if and only if the input I allows for a k-maintainable control, and that the satisfying assignments for $sat(I)$ encode possible such controls.

In our encoding, we shall use for each state $s \in S$ propositional variables s_0, s_1, ..., s_k. Intuitively, s_i will denote that there is a path from state s to some state in E using only agent actions and at most i of them, to which we refer as "*there is an a-path from s to E of length at most i.*"

The encoding $sat(I)$ contains the following formulas:

(0) For all $s \in \mathcal{S}$, and for all j, $0 \leq j < k$:

$$s_j \Rightarrow s_{j+1}$$

(1) For all $s \in E \cap S$:

$$s_0$$

(2) For any two states $s, s' \in \mathcal{S}$ such that $\Phi(a, s) = s'$ for some action $a \in exo(s)$:

$$s_k \Rightarrow s'_k$$

(3) For any state $s \in \mathcal{S} \setminus E$ and all i, $1 \leq i \leq k$:

$$s_i \Rightarrow \bigvee_{s' \in PS(s)} s'_{i-1}, \quad \text{where}$$
$$PS(s) = \{s' \in \mathcal{S} \mid \exists a \in \mathcal{A}_{agent} \cap poss(s) : s' = \Phi(a, s)\};$$

(4) For all $s \in S \setminus E$:

$$s_k$$

(5) For all $s \in \mathcal{S} \setminus E$:

$$\neg s_0$$

The intuition behind the above encoding is as follows. The clauses in (0) state that if there is an a-path from s to E of length at most j then, logically, there is also an a-path of length at most j+1. Next, the clauses in (1) say that for states s in $S \cap E$, there is an a-path of length 0 from s to E. Next, (4) states that for any starting state s in S outside E, there is an a-path from s to E of length at most k, and (5) states that for any state s outside E, there is no a-path from s to E of length 0. The clauses in (3) state that if, for any state s, there is an a-path from s to E of length at most i, then for some possible agent action a and successor state s' = $\Phi(a, s)$, there is an a-path from s' to E of length at most i-1. When looking for k-maintainable controls the clauses in (2) take into account the possibility that s may be in the closure. If indeed s is in the closure and there is an a-path from s to E of length at most k, then the same must be true with respect to the states s' reachable from s using exogenous actions. When looking for super-control they play a role in computing maximal super-controls. The role of each of the above clauses become more clear when relating the models of $sat(I)$ with controls that k-maintain.

Given any model M of $sat(I)$, we can extract a desired control K from it by defining $K(s) = a$ for all s outside E with s_k true in M, where a is a possible agent action in s such that $s' = \Phi(s, a)$ and s' is closer to E than s is. In case of multiple possible a and s', one a can be arbitrarily picked. Otherwise, $K(s)$ is left undefined.

In particular, for $k = 0$, only the clauses from (1), (2), (4) and (5) do exist. As easily seen, $sat(I)$ is satisfiable in this case if and only if $S \subseteq E$ and no exogenous action leads outside E, i.e., the closure of S under exogenous actions is contained in E. This means that no actions of the agent are required at any point in time, and we thus obtain the trivial 0-control K which is undefined on all states, as desired.

The next result states that the SAT encoding works properly in general.

Proposition 5 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ where Φ is deterministic, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, sets of states $E \subseteq \mathcal{S}$ and $S \subseteq \mathcal{S}$, an exogenous function exo, and an integer k. For any model M of $sat(I)$, let $C_M = \{s \in \mathcal{S} \mid M \models s_k\}$, and for any state $s \in C_M$ let $\ell_M(s)$ denote the smallest index j such that $M \models s_j$ (i.e., s_0, s_1,..., s_{j^*-1} are $false$ and s_{j^*} is $true$), which we call the *level* of s w.r.t. M. Then,

(i) S is k-maintainable w.r.t. E iff $sat(I)$ is satisfiable.

(ii) Given any model M of $sat(I)$, the partial function $K^+_M : \mathcal{S} \rightarrow 2^{\mathcal{A}_{agent}}$ defined on $C_M \setminus E$ such that

$$K^+_M(s) = \{a \in \mathcal{A}_{agent} \cap poss(s) \mid \Phi(s,a) = s', \\ s' \in C_M, \ell_M(s') < \ell_M(s)\},$$

is a valid super-control for A w.r.t. $\mathcal{A}_{agent}$;

(iii) any control K which refines K^+_M for some model M of $sat(I)$ k-maintains S w.r.t. E. □

Proof. Since the if direction of (i) follows from (ii) and (iii), it is sufficient to show the only if direction of (i), and then (ii) and (iii).

As for the only if direction of (i), suppose S is k-maintainable w.r.t. E. Then there exists a control K such that for each state $s \in Closure(S, A_{K,exo})$, and for each sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ where $s^{(0)} = s$ in $Unfold_k(s, A, K)$, $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$. We now construct an interpretation M for $sat(I)$ as follows. Since Φ is deterministic, for each s in $Closure(S, A_{K,exo})$ there is a unique sequence $s^{(0)}$ (=s), $s^{(1)}, \ldots, s^{(l)}$ in $Unfold_k(s, A, K)$. Let i (≥ 0) be the smallest index such that $s^{(i)} \in E$. We assign $false$ to s_0, s_1,..., s_{i-1} and assign $true$ to s_i, s_{i+1},..., s_k. All other propositions are assigned $false$. We now argue that M is a model of $sat(I)$.

It is straightforward to see that M satisfies the formulas generated by (0), (1), (4) and (5). Now consider the formulas generated in (2). If s_k is true, then $s \in Closure(S, A_{K,exo})$ by construction. In this case, in order to k-maintain S w.r.t. E, for any $s' = \Phi(a, s)$ of an exogenous action a, one of the states in $Unfold_k(s', A, K)$ must be in E. Hence, s'_k has been assigned true in M. Now let us consider the formulas generated in (3). If s_i is true for some $i \leq k$, then there must be an a-path from s to E of length at most i, emerging from possible agent actions only (via control K). Let s' be the next state in this path. Obviously, there must be an a-path from s' to E of length at most $i-1$ (via K). Hence, s'_{i-1} must be true in M. Thus, M is a model of $sat(I)$, which means that $sat(I)$ is satisfiable.

To show (ii), let us assume that $sat(I)$ has a model M and consider the partial function $K^+_M : \mathcal{S} \rightarrow 2^{\mathcal{A}_{agent}}$ which is defined on $C_M \setminus E$ by $K^+_M(s) = \{a \in \mathcal{A}_{agent} \cap poss(s) \mid \Phi(s,a) = s', s' \in C_M$ and $\ell_M(s') < \ell_M(s)\}$; and for any other s, $K^+_M(s)$ is undefined. For K^+_M to be a valid super-control it must satisfy the following conditions: (a) $K^+_M(s) \subseteq poss(s)$, and (b) $K^+_M(s) \neq \emptyset$ whenever $K^+_M(s)$ is defined. Condition (a) is true by virtue of the construction of K^+_M. Condition (b) is true because $K^+_M(s)$ is defined when $s \in C_M \setminus E$ which means $M \models s_k$ for some $k > 0$, which in turn means that $\ell_M(s) > 0$, thus making $K^+_M(s) \neq \emptyset$.

Now to show (iii), let K be any control which refines K^+_M for some model M of $sat(I)$. Let the distance $d_K(s, S)$ of a state s from the set of states S be the minimum number of transitions – through exogenous actions and/or control actions dictated by the control K – needed to reach s from any state in S.

We will show, by using induction on $d(s, S) \geq 0$, that for every state $s \in Closure(S, A_{K,exo})$ and every sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ with $s = s^{(0)}$ in $Unfold_k(s, A, K)$, the set $\{s^{(0)}, \ldots, s^{(l)}\}$ intersects with E and that $M \models s_k$ (i.e., $s \in C_M$). This proves the claim.

The base case, $d(s, S) = 0$, is about states $s \in S$. From the formulas in (0), (1), and (4) we have $M \models s_k$ for every such state s. Then from the construction of K^+_M above and the formulas in (3), it follows that for any such state s and for every sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ with $s = s^{(0)}$ in $Unfold_k(s, A, K)$, the set $\{s^{(0)}, \ldots, s^{(l)}\}$ intersects with E. Indeed, by taking the action $K(s^{(i)})$ ($\in K^+_M(s^{(i)})$) in $s^{(i)}$, a state $s^{(i+1)} = \Phi(s, K(s^{(i+1)}))$ is reached, such that $\ell_M(s^{(i+1)}) < \ell_M(s^{(i)})$. If $l = k$, then clearly $\ell_M(s^{(l)}) = 0$; otherwise, if $l < k$, then $K(s^{(l)})$ must be undefined, which again implies $\ell_M(s^{(l)}) = 0$. Thus, $s^{(l)} \in E$, which means that $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$.

Thus the statement holds in the base case. Now for the induction step, let us assume that it holds for every state $s \in \mathit{Closure}(S, A_{K,exo})$ at distance $d \geq 0$ from S. Let us now consider a state $s \in \mathit{Closure}(S, A_{K,exo})$ at distance d+1 from S. Then there is a state s' at distance d from S such that $s = \Phi(a, s')$ and either (i) $a \in exo(s')$ or (ii) $a = K(s')$. In both cases, we have by the induction hypothesis that $M \models s'_k$, and using (2), (3), and (1) we can conclude $M \models s_k$; Furthermore, by construction of K and the formulas in (3), we have by similar arguments as above that for each sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ with $s = s^{(0)}$ in $\mathit{Unfold}_k(s, A, K)$, $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$.

This proves our claim. Now each control K as in (ii) is a refinement of K^+_M. This completes the proof.□

5.1.1 Horn SAT encoding

While $sat(I)$ is constructible in polynomial time from I, we can not automatically infer that solving k-MAINTAIN is polynomial, since SAT is a canonical NP-hard problem. However, a closer look at the structure of the clauses in $sat(I)$ reveals that this instance is solvable in polynomial time. Indeed, it is a *reverse Horn* theory; i.e., by reversing the propositions, we obtain a Horn theory. Let us use propositions $\overline{s_i}$ whose intuitive meaning is converse of the meaning of s_i. Then the Horn theory corresponding to $sat(I)$, denoted $\overline{sat}(I)$, is as follows:

(0) For all $s \in \mathcal{S}$ and j, $0 \leq j < k$:

$\overline{s_{j+1}} \Rightarrow \overline{s_j}$.

(1) For all $s \in E \cap S$:

$\overline{s_0} \Rightarrow \bot$.

(2) For any states $s, s' \in \mathcal{S}$ such that $s' = \Phi(a, s)$ for some action $a \in exo(s)$:

$\overline{s'_k} \Rightarrow \overline{s_k}$.

(3) For any state in $\mathcal{S} \setminus E$, and for all i, $1 \leq i \leq k$:

$\left(\bigwedge_{s' \in PS(s)} \overline{s'_{i-1}} \right) \Rightarrow \overline{s_i}$, where

$PS(s) = \{s' \in \mathcal{S} \mid \exists a \in \mathcal{A}_{agent} \cap poss(s) : s' = \Phi(a, s)\}$.

(4) For all $s \in S \setminus E$:

$\overline{s_k} \Rightarrow \bot$.

(5) For all $s \in \mathcal{S} \setminus E$:

$\overline{s_0}$.

Here, $\bot$ denotes falsity. We then obtain a result similar to Proposition 5, and the models M of $\overline{sat}(I)$ lead to k-maintainable controls, which we can construct similarly; just replace in part (ii) C_M with $\overline{C}_M = \{s \in \mathcal{S} \mid M \not\models \overline{s_k}\}$. Notice that $\overline{C}_M$ coincides with the set of states $C_{\overline{M}}$ for the model $\overline{M}$ of $sat(I)$ such that $\overline{M} \models p$ iff $M \not\models \overline{p}$, for each atom p.

We now illustrate the above Horn encoding with respect to an example.

Example 5 Consider the system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, where $\mathcal{S} = \{b, c, d, f, g, h\}$, $\mathcal{A} = \{$ **a**, **a**′, **e**$\}$, and the (deterministic) transition function Φ was shown in Figure 1, where $\Phi(s, a) = s'$ iff an arc $s \rightarrow s'$ labeled with a is present and $poss(s)$ are all actions that label arcs leaving s.

For $\mathcal{A} = \{$ **a**, **a**′ $\}$ and $exo(s) = \{$ **e** $\}$ iff $s = f$ and $exo(s) = \emptyset$ otherwise, this leads for $S = \{b\}$, $E = \{h\}$, and $k = 3$ to the following Horn encoding $\overline{sat}(I)$:

(From 0)

$$\begin{array}{lll} \overline{b_1} \Rightarrow \overline{b_0}. & \overline{b_2} \Rightarrow \overline{b_1}. & \overline{b_3} \Rightarrow \overline{b_2}. \\ \overline{d_1} \Rightarrow \overline{d_0}. & \overline{d_2} \Rightarrow \overline{d_1}. & \overline{d_3} \Rightarrow \overline{d_2}. \\ \overline{g_1} \Rightarrow \overline{g_0}. & \overline{g_2} \Rightarrow \overline{g_1}. & \overline{g_3} \Rightarrow \overline{g_2}. \end{array} \qquad \begin{array}{lll} \overline{c_1} \Rightarrow \overline{c_0}. & \overline{c_2} \Rightarrow \overline{c_1}. & \overline{c_3} \Rightarrow \overline{c_2}. \\ \overline{f_1} \Rightarrow \overline{f_0}. & \overline{f_2} \Rightarrow \overline{f_1}. & \overline{f_3} \Rightarrow \overline{f_2}. \\ \overline{h_1} \Rightarrow \overline{h_0}. & \overline{h_2} \Rightarrow \overline{h_1}. & \overline{h_3} \Rightarrow \overline{h_2}. \end{array}$$

(From 1)

(From 2)

$$\overline{g_3} \Rightarrow \overline{f_3}.$$

(From 3)

$$\begin{array}{rrr} \overline{c_0} \wedge \overline{f_0} \Rightarrow \overline{b_1}. & \overline{c_1} \wedge \overline{f_1} \Rightarrow \overline{b_2}. & \overline{c_2} \wedge \overline{f_2} \Rightarrow \overline{b_3}. \\ \overline{d_0} \Rightarrow \overline{c_1}. & \overline{d_1} \Rightarrow \overline{c_2}. & \overline{d_2} \Rightarrow \overline{c_3}. \\ \overline{h_0} \Rightarrow \overline{d_1}. & \overline{h_1} \Rightarrow \overline{d_2}. & \overline{h_2} \Rightarrow \overline{d_3}. \\ \overline{h_0} \Rightarrow \overline{f_1}. & \overline{h_1} \Rightarrow \overline{f_2}. & \overline{h_2} \Rightarrow \overline{f_3}. \\ \overline{g_1}. & \overline{g_2}. & \overline{g_3}. \end{array}$$

(From 4)

$$\overline{b_3} \Rightarrow \bot.$$

(From 5)

$$\overline{b_0}. \quad \overline{c_0}. \quad \overline{d_0}. \quad \overline{f_0}. \quad \overline{g_0}.$$

This theory has the least model

$$M = \{\overline{g_3}, \overline{g_2}, \overline{g_1}, \overline{g_0}, \overline{f_3}, \overline{f_2}, \overline{f_1}, \overline{f_0}, \overline{b_2}, \overline{b_1}, \overline{b_0}, \overline{c_1}, \overline{c_0}, \overline{d_0}\};$$

hence, $\overline{C}_M = \{b, c, d, h\}$, which gives rise to the super-control K^+ such that $K^+(s) = \{\mathbf{a}\}$ for $s \in \{b, c, d\}$ and $K^+(s)$ is undefined for $s \in \{f, g, h\}$. In this case, there is a single control K refining K^+, which has $K(s) = \mathbf{a}$ for $s \in \{b, c, d\}$ and is undefined otherwise. This is intuitive: The agent must reach h, and has to avoid taking **a**′ in b since then it might arrive at the no-good state g. Thus, she has to take **a** in b and, as the only choice, in the subsequent states c and d. Also, we might not add any state apart from b, c, and d without losing 3-maintainability. In this particular case, M is also maximal on the propositions s_3, where $s \in \mathcal{S} \setminus E = \{b, c, d, f, g\}$: By (4), we can not add $\overline{b_3}$, and by (0) and the clauses $\overline{c_2} \wedge \overline{f_2} \Rightarrow \overline{b_3}$ and $\overline{d_1} \Rightarrow \overline{c_2}$ in (3) then also neither $\overline{c_3}$ nor $\overline{d_3}$. Thus, the above control K is also smallest and, in fact, the only one possible for 3-maintainability. □

As computing a model of a Horn theory is a well-known polynomial problem [16], we thus obtain the following result.

Theorem 6 Under deterministic state transitions, problem k-MAINTAIN is solvable in polynomial time. □

An interesting aspect of the above is that, as well-known, each satisfiable Horn theory T has the least model, M_T, which is given by the intersection of all its models. Moreover, the least model is computable in linear time, cf. [16, 37]. This model not only leads to a k-maintainable control, but also leads to a *maximal* control, in the sense that the control is defined on a greatest set of states outside E among all possible k-maintainable controls for S' w.r.t. E such that $S \subseteq S'$. This gives a clear picture of which other states may be added to S while k-maintainability is preserved; namely, any states in $\overline{C}_{M_T}$. Furthermore, any control K computed from M_T applying the method in Proposition 5 (using $\overline{C}_{M_T}$) works for such an extension of S as well.

On the other hand, intuitively a k-maintainable control constructed from some maximal model of $\overline{sat}(I)$ with respect to the propositions $\overline{s_k}$ is undefined to a largest extent, and works merely for a smallest extension. We may generate, starting from M_T, such a maximal model of T by trying to flip first, step by step all propositions $\overline{s_k}$ which are $false$ to $true$, as well as other propositions entailed. In this way, we can generate a maximal model of T on $\{\overline{s_k} \mid s \in \mathcal{S} \setminus E\}$ in polynomial time, from which a "lean" control can also be computed in polynomial time.

5.2 Non-deterministic transition function $\Phi(s, a)$

We now generalize our method for constructing k-maintainable controls to the case in which transitions due to Φ may be non-deterministic. As before, we first present a general propositional SAT encoding, and then rewrite to a propositional Horn SAT encoding. To explain some of the notations, we need the following definition, which generalizes the notion of an a-path to the non-deterministic setting.

Definition 13 (a-path) We say that there exists an a-path of length at most $k \geq 0$ from a state s to a set of states S', if either $s \in S'$, or $s \notin S'$, $k > 0$ and there is some action $a \in \mathcal{A}_{agent} \cap poss(s)$ such that for every $s' \in \Phi(s, a)$ there exists an a-path of length at most $k - 1$ from s' to S'. □

In the following encoding of an instance I of problem k-MAINTAIN to SAT, referred to as $sat'(I)$, s_i will again intuitively denote that there is an a-path from s to E of length at most i. The proposition s_a_i, $i > 0$, will denote that for such s there is an a-path from s to E of length at most i starting with action a ($\in poss(s)$). The encoding $sat'(I)$ has again groups (0)–(5) of clauses as follows:

(0), (1), (4) and (5) are the same as in $sat(I)$.

(2) For any state $s \in \mathcal{S}$ and s' such that $s' \in \Phi(a, s)$ for some action $a \in exo(s)$:

$s_k \Rightarrow s'_k$

(3) For every state $s \in \mathcal{S} \setminus E$ and for all i, $1 \leq i \leq k$:

(3.1) $s_i \Rightarrow \bigvee_{a \in \mathcal{A}_{agent} \cap poss(s)} s\_a_i$;

(3.2) for every $a \in \mathcal{A}_{agent} \cap poss(s)$ and $s' \in \Phi(s, a)$:

$s\_a_i \Rightarrow s'_{i-1}$;

(3.3) for every $a \in \mathcal{A}_{agent} \cap poss(s)$, if $i < k$:

$s\_a_i \Rightarrow s\_a_{i+1}$.

Group (2) above is very similar to group (2) of $sat(I)$ in the previous subsection. The only change is that we now have $s' \in \Phi(a, s)$ instead of $s' = \Phi(a, s)$. The main difference is in group (3). We now

explain those clauses. The clauses in (3.1) and (3.2) together state that if there is an a-path from s to E of length at most i, then there is some possible action a for the agent, such that for each state s' that potentially results by taking a in s, there must be an a-path from s' to E of length at most i-1. The clauses $s\_a_i \Rightarrow s\_a_{i+1}$ in (3.3) say that on a longer a-path from s the agent must be able to pick a also. Notice that there are no formulas in $sat'(I)$ which forbid to pick different actions a and a' in the same state s, and thus we have a super-control; however, we can always refine it easily to a control.

Proposition 7 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, sets of states $E, S \subseteq \mathcal{S}$, an exogenous function exo, and an integer k. For any model M of $sat'(I)$, let $C_M = \{s \in \mathcal{S} \mid M \models s\_k\}$, and for any state $s \in C_M \setminus E$ let $\ell_M(s)$ denote the smallest index j such that $M \models s\_a_j$ for some action $a \in \mathcal{A}_{agent} \cap poss(s)$, which we call the *a-level* of s w.r.t. M. Then,

(i) S is k-maintainable w.r.t. E iff $sat'(I)$ is satisfiable;

(ii) given any model M of $sat'(I)$, the partial function $K^+_M : \mathcal{S} \rightarrow 2^{\mathcal{A}_{agent}}$ which is defined on $C_M \setminus E$ by

$$K^+_M(s) = \{a \mid M \models s\_a_{\ell_M(s)}\}$$

is a valid super-control; and

(iii) any control K which refines K^+_M for some model M of $sat'(I)$ k-maintains S w.r.t. E.

Proof. The proof follows the line of argumentation in the proof of Proposition 5. It is sufficient to show the only if direction of (i) and both (ii) and (iii).

As for the only if direction of (i), suppose S is k-maintainable w.r.t. E. Then there exists a control K such that for each state $s \in \mathit{Closure}(S, A_{K,exo})$, and for each sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ in $\mathit{Unfold}_k(s, A, K)$ where $s^{(0)} = s$, $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$. We now construct an interpretation M for $sat'(I)$ as follows.

For each $s \in \mathit{Closure}(S, A_{K,exo})$, let in each sequence $\sigma = s^{(0)}$, $s^{(1)}, \ldots$, $s^{(l)}$ in $\mathit{Unfold}_k(s, A, K)$ with $s = s^{(0)}$, the number i_σ (≥ 0) be the smallest index i such that $s^{(i)} \in E$, and let i^* be the maximum over all i_σ for s. Intuitively, i^* is the length of the longest path in the tree with root s where each node n not in E is sprouted by taking the control action $K(n)$ and adding each state in $\Phi(n, K(n))$ as a child. Then, we assign $true$ to s_{i^*}, s_{i^*+1},..., s_k and, if $i^* > 0$, to $s\_a_{i^*}, s\_a_{i^*+1}, \ldots s\_a_k$, where $K(s) = a$. All other propositions are assigned $false$ in M. We now argue that M is a model of $sat(I)$.

It is straightforward to see that M satisfies the formulas generated by (0), (1), (4) and (5). Now consider the formulas $s_k \Rightarrow s'_k$ generated in (2). If s_k is true, then $s \in \mathit{Closure}(S, A_{K,exo})$ by construction. In this case, for any $s' \in \Phi(a, s)$ of an exogenous action a, we have $s' \in \mathit{Closure}(S, A_{K,exo})$, and since K k-maintains S w.r.t. E, s'_i is $true$ in M for some $i \leq k$ which implies, by construction, that s'_k is assigned $true$ in M. Let us finally consider the formulas generated in (3). If s_i, where $s \in \mathcal{S} \setminus E$, is assigned $true$ in M for some $i \in \{1 \leq i \leq k\}$, then $s \in \mathit{Closure}(S, A, K_{exo})$ holds by construction of M. Since K is a k-maintaining control and $s \notin E$, we must have $K(s)$ defined and thus, by construction of M, we have $s\_K(s)_i$ assigned true in M. Since $K(s) \in \mathcal{A}_{agent} \cap poss(s)$, the clause (3.1) is thus satisfied. Furthermore, each clause in (3.2) is satisfied when $a \neq K(s)$, since then s_{a_i} is assigned $false$ in M. For $a = K(s)$, proposition s_{a_i} is $true$ in M and thus, by construction, also s_i. Since K is k-maintaining control, every state $s' \in \Phi(s, a)$ belongs to $\mathit{Closure}(S, A, K_{exo})$. Let, for each sequence $\sigma' = s^{(0)}$, $s^{(1)}, \ldots$, $s^{(l)}$ in $\mathit{Unfold}_k(s, A, K)$ such that $s^{(0)} = s'$, the sequence $P(\sigma) = s^{(0)}$, $s^{(1)}, \ldots, s^{(i)}$ be the shortest prefix of σ such that $s^{(i)} \in E$ (notice that $i < k$). Then, the sequence $s, P(\sigma)$ is a prefix of

some sequence in $Unfold(s, A, K)$. Hence, it follows that in the construction of M, the number i^* for s is larger than the one for s'. Thus, by construction of M, it follows that s'_{i-1} is assigned true in M. This means that the formulas in (3.2) are satisfied in M. Finally, the clauses (3.3) are clearly satisfied in M by construction of M. Thus, M is a model of $sat'(I)$, which means that $sat'(I)$ is satisfiable.

To show (ii), let us assume that $sat'(I)$ has a model M, and consider the partial function $K^+_M : \mathcal{S} \rightarrow 2^{\mathcal{A}_{agent}}$ which is defined on $C_M \setminus E$ by $K^+_M(s) = \{a \mid M \models s\_a_{\ell_M(s)}\}$. We thus have to show that $K^+_M(s) \subseteq poss(s)$ and $K^+_M(s) \neq \emptyset$ when $K^+_M(s)$ is defined. By clause (3.1), and the definition of C_M, ℓ_M, and K^+_M this is immediate.

To show (iii), let K be any control which refines K^+_M for some model M of $sat'(I)$. Let the distance $d_K(s, S)$ of a state s from the set of states S be as in the proof of Proposition 5. i.e., the minimum number of transitions – through exogenous actions and/or control actions dictated by the control K – needed to reach s from any state in S.

We will show, by using induction on $d(s, S) \geq 0$, that for every state $s \in Closure(S, A_{K,exo})$ and every sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ with $s = s^{(0)}$ in $Unfold_k(s, A, K)$, the set $\{s^{(0)}, \ldots, s^{(l)}\}$ intersects with E and that $M \models s_k$ (i.e., $s \in C_M$). This proves that K k-maintains S w.r.t. E.

The base case, $d(s, S) = 0$, is about states $s \in S$. From the formulas in (0), (1), and (4) we have $M \models s_k$ for every such state s. Consider any sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ in $Unfold_k(s, A, K)$ such that $s = s^{(0)}$. If $s \in E$, then we must have $l = 0$, and $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$. Otherwise, $M \models s_{a_k}$ where $a = K(s)$. We then have $s^{(1)} \in \Phi(s, a)$, and thus by our construction of K and the clauses in (3.2) we have that $M \models s^{(1)}_{k-1}$. Repeating this argument, we can infer that $s^{(0)}_k$, $s^{(1)}_{k-1}$, …, $s^{(l)}_{k-l}$ are all assigned $true$ in M. If $k = l$, it follows from the clauses in (5) that $s^{(l)} \in E$. Otherwise, if $l < k$, then K must be undefined on $s^{(l)}$; by the clauses (1), this again means $s^{(l)} \in E$. Hence, $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$.

Thus the statement holds in the base case. Now for the induction step, let us assume that it holds for every state $s \in Closure(S, A_{K,exo})$ at distance $d(s, S) = d \geq 0$ from S. Let us now consider a state $s \in Closure(S, A_{K,exo})$ at distance $d(s', S) = d + 1$ from S. Then there is a state s' at distance $d(s, S) = d$ from S such that $s \in \Phi(a, s')$ and either (i) $a \in exo(s')$ or (ii) $a \in K(s')$. In both cases, we have by the induction hypothesis that $M \models s'_k$, and we can conclude $M \models s_k$ from the clauses in (2) in case (i) and from our construction of K and the clauses in (3.2), (1), and (0) in case (ii), respectively. Furthermore, by similar argumentation as in the case $d = 0$ above, we obtain that for each sequence $\sigma = s^{(0)}, s^{(1)}, \ldots, s^{(l)}$ in $Unfold_k(s, A, K)$ with $s = s^{(0)}$ it holds that $\{s^{(0)}, \ldots, s^{(l)}\} \cap E \neq \emptyset$. This concludes the induction and the proof of (iii). □

One advantage of the encoding $sat'(I)$ over the encoding $sat(I)$ for deterministic transition function Φ above is that it directly gives us the possibility to read off a suitable control from the s_a_i propositions, $a \in poss(s)$, which are true in any model M that we have computed, without looking at the transition function $\Phi(s, a)$ again. On the other hand, the encoding is more involved, and uses a larger set of propositions. Nonetheless, the structure of the formulas in $sat'(I)$ is benign for computation and allows us to compute a model, and from it a k-maintainable control in polynomial time.

5.2.1 Horn SAT encoding (general case)

The encoding $sat'(I)$ is, like $sat(I)$, a reverse Horn theory. We thus can rewrite $sat'(I)$ similarly to a Horn theory, $\overline{sat}'(I)$ by reversing the propositions, where the intuitive meaning of $\overline{s_i}$ and $\overline{s\_a_i}$ is the converse of the meaning of s_i and s_a_i respectively. The encoding $\overline{sat}'(I)$ is as follows:

(0), (1), (4) and (5) are as in $\overline{sat}(I)$

(2) For every states $s, s' \in \mathcal{S}$ such that $s' \in \Phi(a, s)$ for some action $a \in exo(s)$: $\overline{s'_k} \Rightarrow \overline{s_k}$.

(3) For every state $s \in \mathcal{S} \setminus E$ and for all i, $1 \leq i \leq k$:

(3.1) $\left(\bigwedge_{a \in \mathcal{A}_{agent} \cap poss(s)} \overline{s\_a_i} \right) \Rightarrow \overline{s_i}$;

(3.2) for every $a \in \mathcal{A}_{agent} \cap poss(s)$ and $s' \in \Phi(s, a)$:
$\overline{s'_{i-1}} \Rightarrow \overline{s\_a_i}$;

(3.3) for every $a \in \mathcal{A}_{agent} \cap poss(s)$, if $i < k$:
$\overline{s\_a_{i+1}} \Rightarrow \overline{s\_a_i}$.

We obtain from Proposition 7 easily the following result, which is the main result of this section so far.

Theorem 8 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, sets of states $E, S \subseteq \mathcal{S}$, an exogenous function exo, and an integer k. Let, for any model M of $\overline{sat}'(I)$, $\overline{C}_M = \{s \mid M \not\models \overline{s_k}\}$, and let $\overline{\ell}_M(s) = \min\{j \mid M \not\models \overline{s\_a_j}, a \in \mathcal{A}_{agent} \cap poss(s)\}$ for every $s \in \mathcal{S}$. Then,

(i) S is k-maintainable w.r.t. E iff the Horn SAT instance $\overline{sat}'(I)$ is satisfiable;

(ii) Given any model M of $\overline{sat}'(I)$, every control K such that $K(s)$ is defined iff $s \in \overline{C}_M \setminus E$ and satisfies

$$K(s) \in \{a \in \mathcal{A}_{agent} \cap poss(s) \mid M \not\models \overline{s\_a_j}, j = \overline{\ell}_M(s)\},$$

k-maintains S w.r.t. E. □

Corollary 9 Problem k-MAINTAIN is solvable in polynomial time. More precisely, it is solvable in time $O(k\|I\|)$, where $\|I\|$ denotes the size of input I. □

Proof. A straightforward analysis yields that the size of $\overline{sat}'(I)$, measured by the number of atoms in it, is $O(k(|\mathcal{S}| + |\Phi| + |poss|))$, if $\mathcal{A}_{agent}$, S, E, Φ, $poss$ and exo are stored in a standard way as bitmaps, i.e., a (multi-dimensional) array with value range {0,1} (thus, $\|I\| = O(|\mathcal{S}|^2|\mathcal{A}| + \log k)$). Furthermore, the clauses in $\overline{sat}'(I)$ can be easily generated within the same time bound. Since the least model of any Horn theory T is computable in time $O(|T|)$ where $|T|$ is the number of atoms in it [16, 37], deciding satisfiability and computing some model M of $\overline{sat}'(I)$ is feasible in $O(k\|I\|)$ time. Furthermore, $\overline{C}_M$ and $\{(s, \overline{\ell}_M(s)) \mid s \in \mathcal{S}\}$ are computable from M in linear time in the number of atoms, using suitable data structures, and from this a control K as in Theorem 8.(ii) in the same time. Hence, a k-maintaining control for S w.r.t. E is computable in $O(k\|I\|)$ time.

Note that a more economic representation stores S, E, $\mathcal{A}_{agent}$ as sets (i.e., lists) and Φ, $poss$, and exo by their graphs in tables, i.e., sets of tuples $\{\langle s, a, \Phi(s, a)\rangle \mid s \in \mathcal{S}, a \in \mathcal{A}\}$, $\{\langle s, poss(s)\rangle \mid s \in \mathcal{S}\}$, and $\{\langle s, exo(s)\rangle \mid s \in \mathcal{S}\}$. Also under this representation, and if moreover tuples where $\Phi(s, a)=\emptyset$ (resp., $poss(s)=\emptyset$ and $exo(s)=\emptyset$) are not stored (which is of the same order as storing the sets of tuples $\{\langle s, a, s'\rangle \mid s' \in \Phi(a, s)\}$, $\{\langle s, a\rangle \mid a \in poss(s)\}$, $\{\langle s, a\rangle \mid a \in exo(s)\}$), the $O(k\|I\|)$ time bound holds. Indeed, arrays storing S, E, and $\mathcal{A}_{agent}$ for lookup in $O(1)$ time are constructible in time $O(|\mathcal{S}| + |\mathcal{A}|)$. Then, $poss_{agent} = \{\langle s, a\rangle \in poss \mid a \in \mathcal{A}_{agent}\}$ storing $\mathcal{A}_{agent} \cap poss(s)$ for all s is constructible in $O(|poss|)$ time. From this, all clauses of $\overline{sat}'(I)$ except (2) and (3.2) can be readily generated in time

$O(k(|\mathcal{S}| + |poss_{agent}|))$. The clauses (2) and (3.2) can be easily constructed from $\Phi_{exo} = \{\langle s, a, s'\rangle \in \Phi \mid a \in exo(s)\}$ and $\Phi_{poss} = \{\langle s, a, s'\rangle \in \Phi \mid a \in poss(s)\}$ in time $O(|\Phi_{exo}|)$ and $O(k|\Phi_{poss}|)$, respectively. The sets Φ_{exo} and Φ_{poss} can be generated from Φ and exo in time $O(|\Phi| + |exo| + poss|)$, using an auxiliary array $aux[\mathcal{A}, \mathcal{S}]$ to enable random access to $\Phi(a, s)$; notice that $aux[a, s]$ needs not be defined if $\Phi(a, s) = \emptyset$. In total, $\overline{sat}'(I)$ is constructible in $O(|\mathcal{A}| + |exo| + k(|\mathcal{S}| + |\Phi| + |poss|))$ = $O(k\|I\|)$ time. □

Thus in particular, finding a maintaining control under a small window of opportunity, a k-maintaining control for k bounded by a constant, is feasible in *linear time* in the size of the input.

Similar as in Section 5.1.1, the least model of the theory given by $\overline{sat}'(I)$, $M_{\overline{sat}'(I)}$, leads to a *maximal* control in the sense that the pre-image of K outside E, i.e., the states outside E in which K is defined, is greatest among all possible k-maintaining controls which include S. Furthermore, a smallest k-maintaining control can be similarly computed from any maximal model of $\overline{sat}'(I)$ with respect to the propositions $\overline{s_k}$ where s is outside E, which can be generated from $M_{\overline{sat}'(I)}$ by stepwise maximization. Again, both maximal and smallest controls can be computed in polynomial time.

Example 6 Reconsider the system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ from Example 5. Let us modify the transition function Φ such that $\Phi(c, \mathbf{a}) = \{d, f\}$ instead of $\Phi(c, \mathbf{a}) = \{d\}$. Then, for the respective modified instance I of 3-MAINTAIN, denoted I_1, the encoding $\overline{sat}'(I_1)$ looks as follows.

(0), (1), (2), (4), and (5) are as in $sat(I_1)$ in Example 5;

(3.1): $\overline{b\_\mathbf{a}_1} \wedge \overline{b\_\mathbf{a}'_1} \Rightarrow \overline{b_1}.$ $\quad \overline{b\_\mathbf{a}_2} \wedge \overline{b\_\mathbf{a}'_2} \Rightarrow \overline{b_2}.$ $\quad \overline{b\_\mathbf{a}_3} \wedge \overline{b\_\mathbf{a}'_3} \Rightarrow \overline{b_3}.$

$\overline{c\_\mathbf{a}_1} \Rightarrow \overline{c_1}.$ $\quad \overline{c\_\mathbf{a}_2} \Rightarrow \overline{c_2}.$ $\quad \overline{c\_\mathbf{a}_3} \Rightarrow \overline{c_3}.$

$\overline{d\_\mathbf{a}_1} \Rightarrow \overline{d_1}.$ $\quad \overline{d\_\mathbf{a}_2} \Rightarrow \overline{d_2}.$ $\quad \overline{d\_\mathbf{a}_3} \Rightarrow \overline{d_3}.$

$\overline{f\_\mathbf{a}_1} \Rightarrow \overline{f_1}.$ $\quad \overline{f\_\mathbf{a}_2} \Rightarrow \overline{f_2}.$ $\quad \overline{f\_\mathbf{a}_3} \Rightarrow \overline{f_3}.$

$\overline{g_1}.$ $\quad \overline{g_2}.$ $\quad \overline{g_3}.$

(3.2): $\overline{h_0} \Rightarrow \overline{d\_\mathbf{a}_1}.$ $\quad \overline{h_1} \Rightarrow \overline{d\_\mathbf{a}_2}.$ $\quad \overline{h_2} \Rightarrow \overline{d\_\mathbf{a}_3}.$ $\quad \overline{h_0} \Rightarrow \overline{f\_\mathbf{a}_1}.$ $\quad \overline{h_1} \Rightarrow \overline{f\_\mathbf{a}_2}.$ $\quad \overline{h_2} \Rightarrow \overline{f\_\mathbf{a}_3}.$

$\overline{d_0} \Rightarrow \overline{c\_\mathbf{a}_1}.$ $\quad \overline{d_1} \Rightarrow \overline{c\_\mathbf{a}_2}.$ $\quad \overline{d_2} \Rightarrow \overline{c\_\mathbf{a}_3}.$ $\quad \overline{f_0} \Rightarrow \overline{c\_\mathbf{a}_1}.$ $\quad \overline{f_1} \Rightarrow \overline{c\_\mathbf{a}_2}.$ $\quad \overline{f_2} \Rightarrow \overline{c\_\mathbf{a}_3}.$

$\overline{c_0} \Rightarrow \overline{b\_\mathbf{a}_1}.$ $\quad \overline{c_1} \Rightarrow \overline{b\_\mathbf{a}_2}.$ $\quad \overline{c_2} \Rightarrow \overline{b\_\mathbf{a}_3}.$ $\quad \overline{f_0} \Rightarrow \overline{b\_\mathbf{a}'_1}.$ $\quad \overline{f_1} \Rightarrow \overline{b\_\mathbf{a}'_2}.$ $\quad \overline{f_2} \Rightarrow \overline{b\_\mathbf{a}'_3}.$

(3.3): $\overline{d\_\mathbf{a}_2} \Rightarrow \overline{d\_\mathbf{a}_1}.$ $\quad \overline{d\_\mathbf{a}_3} \Rightarrow \overline{d\_\mathbf{a}_2}.$ $\quad \overline{f\_\mathbf{a}_2} \Rightarrow \overline{f\_\mathbf{a}_1}.$ $\quad \overline{f\_\mathbf{a}_3} \Rightarrow \overline{f\_\mathbf{a}_2}.$ $\quad \overline{c\_\mathbf{a}_2} \Rightarrow \overline{c\_\mathbf{a}_1}.$

$\overline{c\_\mathbf{a}_3} \Rightarrow \overline{c\_\mathbf{a}_2}.$ $\quad \overline{b\_\mathbf{a}_2} \Rightarrow \overline{b\_\mathbf{a}_1}.$ $\quad \overline{b\_\mathbf{a}_3} \Rightarrow \overline{b\_\mathbf{a}_2}.$ $\quad \overline{b'\_\mathbf{a}_2} \Rightarrow \overline{b'\_\mathbf{a}_1}.$ $\quad \overline{b'\_\mathbf{a}_3} \Rightarrow \overline{b'\_\mathbf{a}_2}.$

It turns out that $\overline{sat}'(I)$ has no models: From $\overline{g_3}$, the clause $\overline{g_3} \Rightarrow \overline{f_3}$ in (2), and clauses in (0), we obtain that $\overline{f_i}$, $i \in \{0, \ldots, 3\}$, is true in every model M of $\overline{sat}'(I_1)$. Hence, by the clause $\overline{f_2} \Rightarrow \overline{b\_\mathbf{a}_3}$ in (3.2), also $\overline{b\_\mathbf{a}'_3}$ is true in M. On the other hand, from the formula $\overline{f_1} \Rightarrow \overline{c\_\mathbf{a}_2}$ in (3.2), we obtain that $\overline{c\_\mathbf{a}_2}$ must be true in M, and thus by the clauses $\overline{c\_\mathbf{a}_2} \Rightarrow \overline{c_2}$ in (3.1) and $\overline{c_2} \Rightarrow \overline{b\_\mathbf{a}_3}$ in (3.2) that $\overline{b\_a_3}$ is true in M. The clause $\overline{b\_\mathbf{a}_3} \wedge \overline{b\_\mathbf{a}'_3} \Rightarrow \overline{b_3}$ thus implies that $\overline{b_3}$ is true in M. However, by the formula $\overline{b_3} \Rightarrow \bot$ in (4), $\overline{b_3}$ must be false in M. Thus, no model M of $\overline{sat}'(I_1)$ can exist, which by Theorem 8 means that there is no 3-maintaining control for $S = \{b\}$ w.r.t $E = \{h\}$. Indeed, regardless of whether a control function K selects $\mathbf{a}$ or $\mathbf{a}'$ in state b, within at most 2 steps from b the state f might be reached, from which the exogenous function might move the system to the no-good state g.

Suppose now again that $\Phi(c, \mathbf{a}) = \{d, f\}$ and that the agent can take $\mathbf{a}'$ in g, which results in either h or f (i.e., $\Phi(g, \mathbf{a}') = \{f, h\}$ and $\mathbf{a}' \in poss(g)$). Then the Horn encoding $\overline{sat}'(I_1)$ changes as follows:

In (3.1), the facts $\overline{g_i}$, $i \in \{1, 2, 3\}$, are replaced by $\overline{g\_\mathbf{a}_i} \Rightarrow \overline{g_i}$;

In (3.2.), the clauses for $\mathbf{a}'$ and f, h are added, $i \in \{1, 2, 3\}$:

$$\overline{f_0} \Rightarrow \overline{g\_\mathbf{a}'_1}. \qquad \overline{f_1} \Rightarrow \overline{g\_\mathbf{a}'_2}. \qquad \overline{f_2} \Rightarrow \overline{g\_\mathbf{a}'_3}. \qquad \overline{h_0} \Rightarrow \overline{g\_\mathbf{a}'_1}. \qquad \overline{h_1} \Rightarrow \overline{g\_\mathbf{a}'_2}. \qquad \overline{h_2} \Rightarrow \overline{g\_\mathbf{a}'_3}.$$

In (3.3), the clauses for $\mathbf{a}'$ and g are added:

$$\overline{g\_\mathbf{a}'_2} \Rightarrow \overline{g\_\mathbf{a}'_1}. \qquad \overline{g\_\mathbf{a}'_3} \Rightarrow \overline{g\_\mathbf{a}'_2}.$$

In this encoding $\overline{sat}'(I_2)$ of the modified instance I_2, we now longer have a fact $\overline{g_3}$ in (3.1.) and thus the above derivation of a contradiction for the truth value of $\overline{b_3}$ in any model of $\overline{sat}'(I_2)$ is not applicable. In fact, $\overline{sat}'(I_2)$ is satisfiable, and its least model is

$$M \;=\; \{\overline{b_0},\, \overline{c_0},\, \overline{d_0}\,\overline{f_0},\, \overline{g_0}, \overline{b\_\mathbf{a}_1},\, \overline{c\_\mathbf{a}_1},\, \overline{b\_\mathbf{a}'_1},\, \overline{g\_\mathbf{a}'_1},\, \overline{b_1},\, \overline{c_1},\, \overline{g_1},\, \overline{b\_\mathbf{a}_2}\}.$$

Then, we have $\overline{C}_M = \{b, c, d, f, g, h\}$, $\overline{\ell}_M(b) = \overline{\ell}_M(c) = \overline{\ell}_M(g) = 2$ and $\overline{\ell}_M(d) = \overline{\ell}_M(f) = 1$, which leads to a single 3-maintaining control K such that $K(s)$ = **a** for $s \in \{b, c, d, f\}$ and $K(g)$= $\mathbf{a}'$. Note that since K is defined on every state except h, it 3-maintains every set S w.r.t. every E which includes h. As for $S = \{b\}$, $K(c)$ and $K(d)$ could remain undefined, since they are not in the closure of b (which can be easily detected) at the price of losing robustness with respect to enlarging S. There is an alternative solution in which $K(b)$ = $\mathbf{a}'$ instead of $K(b)$ = **a**. Here $K(s)$ can not be made undefined on any $s \neq h$.□

5.3 Genuine algorithm

From the encoding to Horn SAT above, we can distill a direct algorithm to construct a k-maintainable control, if one exists. The algorithm mimics the steps which a SAT solver might take in order to solve $sat'(I)$. It uses counters $c[s]$ and $c[s\_a]$ for each state $s \in \mathcal{S}$ and possible agent action a in state s, which range over $\{-1, 0, \ldots, k\}$ and $\{0, 1, \ldots, k\}$, respectively. Intuitively, value i of counter $c[s]$ (at a particular step in the computation) represents that so far $\overline{s_0}, \ldots, \overline{s_i}$ are assigned true; in particular, $i = -1$ represents that no $\overline{s_i}$ is assigned true yet. Similarly, value i for $c[s\_a]$ (at a particular step in the computation) represents that so far $\overline{s\_a_1}, \ldots, \overline{s\_a_i}$ are assigned true (and in particular, $i = 0$ that no $\overline{s\_a_i}$ is assigned true yet).

Starting from an initialization, the algorithm updates by demand of the clauses in $\overline{sat}'(I)$ the counters (i.e., sets propositions true) using a command $upd(c, i)$ which is short for "if $c < i$ then $c := i$," towards a fixpoint. If a counter violation is detected, corresponding to violation of a clause $\overline{s_0} \rightarrow \bot$ for $s \in S \cap E$ in (1) or $\overline{s_k} \rightarrow \bot$ for $s \in S \setminus E$ in (4), then no control is possible. Otherwise, a control is constructed from the counters.

In detail, the algorithm is as follows:

Algorithm k-CONTROL

Input: A system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions, sets of states $E, S \subseteq \mathcal{S}$, an exogenous function exo, and an integer $k \geq 0$.

Output: A control K which k-maintains S with respect to E, if any such control exists. Otherwise, output that no such control exists.

(Step 1) Initialization

(i) Set $\Phi_{exo} = \{\langle s, a, s'\rangle \mid s \in \mathcal{S},\ a \in exo(s), s' \in \Phi(s, a)\}$, $\Phi^{\overline{E}}_{poss} = \{\langle s, a, s'\rangle \mid s \in \mathcal{S}\backslash E,\ a \in poss(s), s' \in \Phi(s, a)\}$, and for every $s \in \mathcal{S}$, $poss_{ag}(s) = \mathcal{A}_{agent} \cap poss(s)$.

(ii) For every s in E, set $c[s] := -1$.

(iii) For every s in $\mathcal{S} \setminus E$, set $c[s] := k$ if $s \in S$ and $poss_{ag}(s) = \emptyset$; otherwise, set $c[s] := 0$.

(iv) For every s in $\mathcal{S} \setminus E$ and $a \in poss_{ag}(s)$, set $c[s\_a] := 0$.

(Step 2) Repeat the following steps until there is no change or $c[s]{=}k$ for some $s \in S \setminus E$ or $c[s]{\geq}0$ for some $s \in S \cap E$:

(i) For any $\langle s, a, s'\rangle \in \Phi_{exo}$ such that $c[s']{=}k$ do $upd(c[s], k)$.

(ii) For any $\langle s, a, s'\rangle \in \Phi^{\overline{E}}_{poss}$ such that $c[s']{=}i$ and $0 \leq i < k$ do $upd(c[s\_a], i + 1)$.

(iii) For any state $s \in \mathcal{S} \setminus E$ such that $poss_{ag}(s) \neq \emptyset$ and $i{=}\ \min(c[s\_a] \mid a \in poss_{ag}(s))$ do $upd(c[s], i)$.

(Step 3) If $c[s]{=}k$ for some $s \in S \setminus E$ or $c[s]{\geq}0$ for some $s \in S \cap E$, then output that S is not k-maintainable w.r.t. E and halt.

(Step 4) Output any control $K : \mathcal{S} \setminus E \to \mathcal{A}_{agent}$ defined on all states $s \in \mathcal{S} \setminus E$ with $c[s] < k$ and such that $K(s) \in \{a \in poss_{ag}(s) \mid c[s\_a] = \min_{b \in poss_{ag}(s)} c[s\_b] < k\}$. □

The above algorithm is easily modifiable if we simply want to output a super-control such that each of its refinements is a k-maintainable control, leaving a choice about the refinement to the user. Alternatively, we can implement in Step 4 such a choice based on preference information.

The following proposition states that the algorithm works correctly and runs in polynomial time.

Proposition 10 Algorithm k-CONTROL solves problem k-MAINTAIN, and terminates for any input I in polynomial time. Furthermore, it can be implemented to run in $O(k\|I\|)$ time.

Proof. The correctness of the algorithms follows from Theorem 8 and the fact that k-CONTROL mimics, starting from facts in (5) and (3.1), the computation of the least model of $\overline{sat}'(I)$ by a standard fix-point computation. As for the polynomial time complexity, since counters are only increased, and the loop in Step 2 is reentered only if at least one counter has increased in the latest run, it follows that the number of iterations is polynomially bounded. Since the body of Step 2 and each other step is polynomial, it follows that k-CONTROL runs in polynomial time.

For the more detailed account, note that bitmaps for S, E and $\mathcal{A}$ (if not available in the input) can be generated in time $O(|\mathcal{S}| + |\mathcal{A}|)$. In (i) of Step 1, the sets Φ_{exo} and $\Phi^{\overline{E}}_{poss}$ can be constructed in time $O(|\Phi| + |exo|)$ and $O(|\Phi| + |poss| + |\mathcal{S}|)$, respectively, using an auxiliary array for random access to $\Phi(a, s)$ in case if the functions are given by their graphs (cf. proof of Corollary 9). Constructing $poss_{ag}(s)$ for all $s{\in}\mathcal{S}$ takes $O(|poss|)$ time, and (ii)–(iv) of Step 1 is feasible in time $O(|\mathcal{S}| + |poss|)$.

Using flags to signal changes to counters $c[s]$, $c[s_a]$, and auxiliary counters for $\min(c[s\_a] \mid a \in poss_{ag}(s))$, the number of calls of upd in Step 2 is $O(k(|\Phi_{exo}| + |\Phi_{poss}| + |\mathcal{S}|))$, and each call takes $O(1)$ time. The loop condition can be checked in $O(m)$ time where m is the number of changes in the loop. Hence, the total time for Step 2 is $O(k\|I\|)$. Step 3 is $O(1)$ if a flag is set in Step 2 indicating the reason for the loop exit. Finally, in Step 4, a control K can be easily output in time $O(|poss|)$. In total, the time is $O(k\|I\|)$ □

Thus, for k bounded by a constant, k-CONTROL can be implemented to run in linear time. We remark that further improvements are possible. For example, states may be eliminated beforehand which will not be reachable from any state in S under any control that is eventually constructed. This can be done efficiently by computing an upper bound of $Closure(S, K_{A,exo})$ in which all possible actions at any state are merged into a single action. We leave a detailed discussion of this and further refinements for future work.

6 Encoding Maintainability for an Answer Set Solver

In this section, we use the results of the previous section to show how computing a k-maintainable control can be encoded as finding answer sets of a non-monotonic logic program. More precisely, we describe an encoding to non-monotonic logic programs under the Answer Set Semantics [24], which can be executed on one of the available Answer Set Solvers such as DLV [20, 33] or Smodels [40, 50]. These solvers support the computation of answer sets (models) of a given program, from which solutions (in our case, k-maintaining controls) can be extracted.

The encoding is generic, i.e., given by a *fixed program* which is evaluated over the instance I represented by input facts $F(I)$. It makes use of the fact that non-monotonic logic programs can have multiple models, which correspond to different solutions, i.e., different k-maintainable controls.

In the following, we first describe how a system is represented in a logic program, and then we develop the logic programs for both deterministic and general, nondeterministic domains. We shall follow here the syntax of the DLV system; the changes needed to adapt the programs to other Answer Set Solvers such as Smodels are very minor.

6.1 Input representation

The input I of problem k-MAINTAIN, can be represented by facts $F(I)$ as follows.

- The system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ can be represented using predicates `state`, `transition`, and `poss` by the following facts:
  - `state(`s`)`, for each $s \in \mathcal{S}$;
  - `action(`a`)`, for each $a \in \mathcal{A}$;
  - `transition(`s`,`a`,`s'`)`, for each $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ such that $s' \in \Phi(s, a)$;
  - `poss(`s`,`a`)`, for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$ such that $a \in poss(s)$.
- the set $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions is represented using a predicate `agent` by facts `agent(`a`)`, for each $a \in \mathcal{A}_{agent}$;
- the set of states S is represented by using a predicate `start` by facts `start(`s`)`, for each $s \in S$;
- the set of states E is represented by using a predicate `goals` by facts `goal(`s`)`, for each $s \in E$;
- the exogenous function exo is represented by using a predicate `exo` by facts `exo(`s`,`a`)` for each $s \in \mathcal{S}$ and $a \in exo(s)$.
- finally, the integer k is represented using a predicate `limit` by the fact `limit(`k`)`.

Example 7 Coming back to Example 3, the input I is represented as follows:

```
state(b). state(c). state(d). state(f). state(g). state(h).
action(a). action(a1). action(e).
trans(b,a,c). trans(b,a1,f). trans(c,a,d). trans(d,a,h).
trans(f,a,h). trans(f,e,g).
poss(b,a). poss(b,a1). poss(f,a). poss(f,e).
poss(c,a). poss(d,a).
agent(a). agent(a1).
start(b). goal(h).
exo(f,e).
limit(3).
```

□

6.2 Deterministic transition function Φ

The following is a program, executable on the DLV engine, for deciding the existence of a k-control. In addition to the predicates for the input facts $F(I)$, it employs a predicate `n_path(X,I)`, which intuitively corresponds to $\overline{X_I}$, and further auxiliary predicates.

```
% Define range of 0,1,...,k for stages.
 range(I) :- #int(I), I <= K, limit(K).

% Rule for (0).
 n_path(X,I) :- state(X), range(I), limit(K), I<K, n_path(X,J), J = I+1.


% Rule for (1).
 :- n_path(X,0), goal(X), start(X).


% Rule for (2)
 n_path(X,K) :- trans(X,A,Y), exo(X,A), n_path(Y,K), limit(K).


% Rules for (3)
 n_path(X,I) :- state(X), not goal(X), range(I), I>0, not some_pass(X,I).
 some_pass(X,I) :- range(I), I>0, trans(X,A,Y), agent(A),
                   poss(X,A), not n_path(Y,J), I=J+1.


% Rule for (4)
 :- n_path(X,K), limit(K), start(X), not goal(X).


% Rule for (5)
 n_path(X,0) :- state(X), not goal(X).
```

The predicate `range(I)` specifies the index range from 0 to k, given by the input `limit`(k). The rules encoding the clause groups (0) – (2) and (4), (5) are straightforward and self explanatory. For (3), we need to encode rules with bodies of different size depending on the transition function Φ, which itself is part of the input. We use that the antecedent of any implication (3) is true if it is not falsified,

where falsification means that some atom $\overline{s'_{i-1}}$, $s' \in PS(s)$, is false; to assess this, we use the auxiliary predicate `some_pass(X,I)`.

To compute the super-control K^+, we may add the rule:

```
% Define C̄_M
 cbar(X) :- state(X), not n_path(X,K), limit(K).

%Define state level L
 level(X,I) :- cbar(X), not n_path(X,I), I > 0,  n_path(X,J), I=J+1.

 level(X,0) :- cbar(X), not n_path(X,0).

% Define super-control k_plus
 k_plus(X,A) :- agent(A), trans(X,A,Y), poss(X,A), level(X,I),
                level(Y,J), J<I, not goal(X).
```

In `cbar(X)`, we compute the states in $\overline{C}_M$, and in `level(X,I)` the level $\ell_{\overline{M}}(s)$ of each state $s \in \overline{C}_M$ (=$C_{\overline{M}}$ for the corresponding model $\overline{M}$ of $sat(I)$). The super-control K^+_M is then computed in `k_plus(X,A)`.

Finally, by the following rules we can nondeterministically generate any control which refines K^+_M:

```
% Selecting a control from k_plus.
 control(X,Y) :- k_plus(X,Y), not exclude_k_plus(X,Y).

 exclude_k_plus(X,Y) :- k_plus(X,Y), control(X,Z), Y<>Z.
```

The first rule enforces that any possible choice for $K(s)$ must be taken unless it is excluded, which by the second rule is the case if some other choice has been made. In combination the two rules effect that one and only one element from $K^+_M(s)$ is chosen for $K(s)$.

Example 8 If the input representation of Example 5 is in a file `exa3.dlv` and the above program, denoted by Π_{det}, in a file `det.dlv`, the DLV engine can be invoked e.g. by

```
dlv exa3.dlv det.dlv -N=3 -filter=control
```

which outputs the controls; here `-N=3` sets the range of integers dynamically supported by the engine to 3, and -filter=control effects that the answer sets are clipped to the predicate `control`. In the particular case, the output on the call is (apart from system version information)

```
control(b,a), control(c,a), control(d,a)
```

yielding the unique control which exists in this case. If we would add a further agent action $\mathbf{a}_2$ to the action set, and extend the transition function by $\Phi(b, \mathbf{a}_2) = c$, then a call of DLV for the respective representation would yield

```
{control(b,a2), control(c,a), control(d,a)}
{control(b,a), control(c,a), control(d,a)}
```

corresponding to the two alternative controls which emerge, since the agent can take either action **a** or action $\mathbf{a}_2$ in state a.

6.3 Nondeterministic transition function Φ

As for deciding the existence of a k-maintaining control, the only change in the code for the deterministic case affects Step (3). The modified code is as follows, where `n_apath(X,A,I)` intuitively corresponds to $\overline{X\_A_I}$.

```
% Rules for (3); different from above
% (3.1)
 n_path(X,I) :- state(X), not goal(X), range(I), I>0, not some_apass(X,I).

 some_apass(X,I) :- range(I), I>0, agent(A), poss(X,A), not n_apath(X,A,I),
                    not goal(X).

% (3.2)
 n_apath(X,A,I) :- agent(A), trans(X,A,Y), poss(X,A), range(I), I>0,
                   n_path(Y,J), I=J+1, not goal(X).

% (3.3)
 n_apath(X,A,I) :- agent(A), poss(X,A), range(I), I>0, limit(K), I<K,
                   n_apath(X,A,J), J=I+1, not goal(X).
```

Here, `some_apass(X,A,I)` plays for encoding (3.1) a similar role as `some_pass(X,I)` for encoding (3) in the deterministic encoding.

To compute the super-control K^+_M, we may then add the following rules:

```
% Define C̄_M
 cbar(X) :- state(X), not n_path(X,K), limit(K).

% Define state action level, alevel (>=1)
 alevel(X,I) :- alevel_leq(X,I), I=J+1, range(J), not level_leq(X,J).

 alevel_leq(X,I) :- cbar(X), not goal(X), poss(X,A), agent(A), I>0,
                    range(I), not n_apath(X,A,I).

% Define super-control k_plus
 k_plus(X,A) :- agent(A), alevel(X,I), poss(X,A), not n_apath(X,A,I).
```

Here, the value of $\overline{\ell}_M(s)$ is computed in `alevel(X,I)`, using the auxiliary predicate `alevel_leq(X,I)` which intuitively means that $\overline{\ell}_M(X) \leq I$.

For computing the controls refining K^+_M, we can add the two rules for selecting a control from `k_plus` from the program for the deterministic case.

Example 9 Let us revisit the instance I_1 in Example 6. We get the DLV representation of I_1 by adding the fact `trans(c,a,f).` to the representation for I. Assuming that it is in a file `exa4.dlv` and the program Π_{ndet} in a file `ndet.dlv`, a call

```
dlv exa4.dlv ndet.dlv -N=3 -filter=control
```

yields no output (apart from some system version print), which is correct. On the other hand, if we consider the input I_2 for the variant of Example 6 (with agent action $\mathbf{a}'$ possible in g and $\Phi(g, \mathbf{a}') = \{f, h\}$), then the output is

```
{control(b,a1), control(c,a), control(d,a), control(f,a), control(g,a1)}
{control(b,a), control(c,a), control(d,a), control(f,a), control(g,a1)}
```

(where `a1` encodes $\mathbf{a}'$). Again, this is the correct result.

6.4 Layered use of negation

An important note at this point is that the programs Π_{det} and Π_{ndet} do not necessarily have models which correspond to the least models of the Horn theories $\overline{sat}(I)$ and $\overline{sat}'(I)$, respectively. The reason is that the use of negation `not some_pass(X,I)` and resp. `not some_apass(X,I)` may lead through cycles in recursion. Thus, not each control computed is necessarily maximal (even though the maximal controls will be computed in some models). Furthermore, because of cyclic negation it is not a priori clear that the part of the program deciding the existence of a control is evaluated by DLV in polynomial time. However, consistency (i.e., existence of an answer set) is guaranteed whenever $\overline{sat}(I)$ resp. $\overline{sat}'(I)$ has a model.

It is possible to modify Π_{det} such that the use of negation in recursion cycles is eliminated, by using standard coding methods to evaluate the body of the rule in (3). Namely, introduce for Π_{det} a predicate `all_true` and replace `not some_pass(X,I)` in the code for (3) with `all_true(X,I)`, which is defined such that `all_true`(s, i) represents that every $\overline{s'}_{i-1} \in PS(s)$ is assigned true, which can be checked using a linear ordering $\leq$ on $PS(s)$. However, we refrain from this here.

Notice that in the case where $PS(s)$ has size bounded by a constant c, we can use a predicate `ps` of arity $c + 1$ to represent $PS(s) = \{s^{(1)}, \ldots, s^{(l)}\}$ by a single fact `ps`$(s, s^{(1)}, \ldots, s^{(l)}, \ldots, s^{(l)})$ where $s^{(l)}$ is reduplicated if $l < c$. It is then easy to express the clause (3).

We can similarly modify Π_{ndet} such that the use of negation in recursion cycles is eliminated, where we use a linear ordering on $\mathcal{A}_{agent} \cap poss(s)$ (or simply on $\mathcal{A}_{agent}$, assuming that there are not many agent actions overall). Finally, we can also use for the program Π_{det} simply an ordering of $\mathcal{A}_{agent}$, since the deterministic transformation $\Phi(s, a)$ is a (partial) surjective mapping of $\mathcal{A}$ onto $PS(s)$, which guarantees that via $\mathcal{A} \cap poss(s)$ each $s' \in PS(s)$ can be accessed through Φ.

The modified programs use negation only in a stratified manner, and thus will be evaluated by DLV in guaranteed polynomial time in the size of the DLV representation of $\overline{sat}(I)$ and $\overline{sat}'(I)$, respectively.

6.5 State descriptions by variables

In many cases, states of a system are described by a vector of values for parameters which are variable over time. It is easy to incorporate such state descriptions into the LP encoding from above, and to evaluate them on Answer Set Solvers provided that the variables range over finite domains. In fact, if any state s is given by a (unique) vector $s = \langle s^1, \ldots, s^m \rangle$ $m > 0$, of values s^i, $1 \leq i \leq m$, for variables X_i ranging over nonempty domains, then we can represent s as fact `state(`v^i_1`,...,`$v^i_{r_i}$`)` and use a vector `X1,...,Xm` of state variables in the DLV code, in place of a single variable, `X`. No further change of the programs from above is needed.

Similarly, we can easily accommodate actions $a(P_1, P_2, \ldots, P_m)$ with parameters $P_1, \ldots, P_m$ (which is important) from a finite set if desired. However, here rule the defining `exclude_k_plus(X,Y)`

should be replaced by all rules emerging if the atom `Y <> Z` in the body is replaced by `Yi <> Zi`, `i` $\in \{$`1,...,m`$\}$ (assuming that `Y` and `Z` are replaced by `Y1,...,Ym` and `Z1,...,Zm`, respectively).

Another possibility to handle state descriptions by variables would be to implement a coding scheme, which maps each vector $s = \langle s^1, \ldots, s^m \rangle$ into an integer $i(s)$, represented by fact `code`$(i(s), s^1, \ldots, s^m)$.

Furthermore, we point out that the input need not consist merely of facts, but may also involve rules to define the predicates of the input representation more compactly. Finally, the facts for `action` can be dropped, since they are not referenced by any rule in programs Π_{det} and Π_{ndet}.

For illustration, we consider the buffer example from Section 3.

Example 10 Recall that states in the buffer example are given by pairs of integers $\langle i,j \rangle$ where i and j are the numbers of objects in buffer b_1 and b_2, respectively. We thus use variables `X1,X2` and `Y1,Y2` in place of `X` and `Y`, respectively.

For buffer capacity of 3, $S = \{\langle 0,0 \rangle\}$, $E = \{\langle 0,j \rangle \mid 1 \leq j \leq 3\}$, and $k = 6$, the input can be represented as follows:

```
state(X1,X2) :- #int(X1), #int(X2), X1 <= 3, X2 <= 3.

start(0,0).

goal(0,X2) :- state(0,X2).

trans(X1,X2,m_12,Y1,Y2) :- state(X1,X2), state(Y1,Y2), X1=Y1+1, Y2=X2+1.
trans(X1,X2,m_21,Y1,Y2) :- state(X1,X2), state(Y1,Y2), Y1=X1+1, X2=Y2+1.
trans(X,X2,proc,X,Y2) :- state(X,X2), state(X,Y2), X2=Y2+1.
trans(X1,X,ins,Y1,X) :- state(X1,X), state(Y1,X), Y1=X1+1.

poss(X1,X2,m_12) :- state(X1,X2), 1 <= X1, X2 <= 2.
poss(X1,X2,m_21) :- state(X1,X2), 1 <= X2, X1 <= 2.
poss(X1,X2,proc) :- state(X1,X2), 1 <= X2.
poss(X1,X2,ins) :- state(X1,X2), X1 <= 2.

agent(m_12). agent(m_21). agent(proc). exo(ins).

limit(6).
```

Here, equalities `X1=0` for `X1,X2` in the rule defining `goal` and `X1=Y1` in the definition of `trans(X,X2,proc,X,Y2)` etc are pushed through.

Invoking DLV, assuming the representation is stored in file `exa-buffer.dlv` and the expanded version of Π_{det} in a file `det2.dlv`, with

```
dlv exa-buffer.dlv det2.dlv -N=6 -filter=control
```

yields 13 models, of which encode different controls. Among the maximal controls is

```
{ control(1,0,m_12), control(1,1,m_12), control(1,2,m_12), control(1,3,proc),
  control(2,0,m_12), control(2,1,m_12), control(2,2,proc), control(2,3,proc),
  control(3,0,m_12), control(3,1,proc), control(3,2,proc), control(3,3,proc)}
```

which is defined on all states outside E, and thus constitutes a 6-maintaining control for the whole system.

7 Computational Complexity

In this section, we consider the complexity of constructing k-maintainable controls under various assumptions. To this end, we first describe the problems analyzed and give an overview of the complexity results. After that, the results are established in a separate subsection; the reader who is not interested in the technical proofs might safely skip it.

7.1 Problems considered and overview of results

Following the common practice, we consider here the decision problem associated with k-MAINTAIN, which we refer to as k-MAINTAINABILITY: Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$ of agent actions, sets of states $E, S \subseteq \mathcal{S}$, an exogenous function exo, and an integer $k \geq 0$, decide whether S is k-maintainable with respect to E in A. Furthermore, we also consider ω-MAINTAINABILITY, which has the same input except k and asks whether S is maintainable with respect to E in A.

We consider the problems in two different input settings, in line with the previous sections:

Enumerative representation: The constituents of an instance I are explicitly given, i.e., the sets ($\mathcal{A}, \mathcal{S}, \mathcal{A}_{agent}, S$, and E) in enumerative form and the functions ($\Phi(a, s)$, $poss(s)$, and exo) by their graphs in tables.

State variables representation: A system state s is represented by a vector $s = (v_1, \ldots, v_m)$ of values for variables f_1,...,f_m ranging over given finite domains $D_1, \ldots, D_m$, while $\mathcal{A}$ and $\mathcal{A}_{agent}$ are given in enumerative form. We assume that polynomial-time procedures for evaluating the following predicates are available:

- $in\_Phi(s, a, s')$, $in\_poss(s, a)$, and $in\_exo(s, a)$ respectively for deciding $s' \in \Phi(s, a)$, $a \in poss(s)$, and $a \in exo(s)$, respectively.
- $in\_S(s)$ and $in\_E(s)$ for deciding whether $s \in S$ and $s \in E$, respectively.

Orthogonal to this, we also consider (1) general k versus constant k, in order to highlight the complexity of small windows of opportunity for maintenance; (2) absence of exogenous actions, to see what cost intuitively is caused by an adversary; and (3) nondeterministic versus deterministic actions.

The results of the complexity analysis are compactly summarized in Tables 1 and 2, in which unless stated otherwise, the entries stand for completeness results under logspace reductions. We assume that the reader is familiar with the classes **P** (polynomial time), **EXP** (exponential time), **L** (logarithmic workspace), **NL** (nondeterministic logarithmic work space), co-**NP** (co-nondeterministic polynomial time), and **PSPACE** (polynomial space) appearing in the tables, and refer to [44] and references therein for further background on complexity. By **LH** we denote the logarithmic time hierarchy [7, 27], which is given by **LH** = $\bigcup_{i \geq 0} \Sigma_i^{\log}$, where $\Sigma_i^{\log}$ denotes the decision problems solvable on an alternating Turing machine in logarithmic time with at most $i-1$ alternations between existential and universal states, starting in an existential state. Note that **LH** is strictly included in **L**. A more refined complexity assessment is given in Section 7.2. However, we refrain here from providing a sharp complexity characterization of

+/- exogenous actions	k-MAINTAINABILITY		ω-MAINTAINABILITY
	given k	constant $k \geq 1$	
deterministic	**P** / **NL** (Th.11/15)	**P** / in **LH** ($\subset$ **L**) (Th.11/16)	**P** / **NL** (Co.12/Th.15)
nondeterministic	**P** (Th.11/13)	**P** / in **LH** ($\subset$ **L**) (Th.11/16)	**P** (Co.12/Th.13)

Table 1: Complexity of deciding k- and ω-MAINTAINABILITY under enumerative representation (logspace completeness)

+/- exogenous actions	k-MAINTAINABILITY		ω-MAINTAINABILITY
	given k	constant $k \geq 1$	
deterministic	**EXP** / **PSPACE** (Th.18/21)	**EXP** / co-**NP** (Th.18/22)	**EXP** / **PSPACE** (Co.19/Th.21)
nondeterministic	**EXP** (Th.18/20)	**EXP** / co-**NP** (Th.18/22)	**EXP** (Co.19/Th.20)

Table 2: Complexity of deciding k- and ω-MAINTAINABILITY under state variables representation (logspace completeness)

the problems classified within **LH** in terms of completeness under a suitable notion of reduction, since they are not central to the maintainability issue under an "adversarial" environment.

Under enumerative representation (Table 1), k- and ω-MAINTAINABILITY have the same complexity as Horn SAT, which is **P**-complete [44]. Thus, according to widely believed complexity hypotheses, the problem is difficult to parallelize and to solve within poly-logarithmic workspace. In fact, this holds also for the case of constant $k = 1$ and the restriction that all actions are deterministic and that there is a single exogenous action. Thus, even in the simplest setting with an adversary according to the dimensions above, the problem already harbors its full complexity; excluding nondeterministic actions and/or fixing k does not make the problems simpler. Intuitively, this is because with the help of exogenous actions, one can simulate nondeterminism and split sequences of agent maintenance actions into small segments.

On the other hand, when exogenous actions are excluded (listed under "-"), k- and ω-MAINTAINABILITY are always easier when the actions are deterministic or the window of opportunity is small (k is constant). In summary, the results show that exogenous actions can *not* be compiled efficiently away (with reasonable complexity) to an instance of maintainability under a small window opportunity, and that nondeterministic actions are indispensable for such a compilation.

The reason is that in absence of exogenous actions, k-MAINTAINABILITY is akin to a graph reachability resp. planning problem (for the latter, see Section 8.1). Indeed, define for a fixed system $A{=}(\mathcal{S}, \mathcal{A}, \Phi, \mathit{poss})$, a set of agent action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and sets $E, S \subseteq \mathcal{S}$ of states the predicates $r_i(s)$, $i \geq 0$, on $s \in \mathcal{S}$ inductively by

$$\begin{aligned} r_0(s) &= s \in E, \\ r_{i+1}(s) &= s \in E \vee \exists a \in \mathcal{A}_{agent} \cap \mathit{poss}(s) \\ &\qquad \forall s' \in \mathcal{S}(s' \in \Phi(s,a) \Rightarrow r_i(s')), \qquad \text{for } i \geq 0. \end{aligned} \tag{1}$$

Informally, $r_i(s)$ expresses that some state in E can be reached from s within i agent actions, and it holds that S is k-maintainable with respect to E, exactly if $r_k(s)$ holds for every s in S (as proved in Lemma 1 below). The predicate $r_k(s)$ is definable in first-order predicate logic with a suitable relational vocabulary (using the predicates given for enumerative representation). As well-known, the first-order

definable properties are those which can be decided in **LH** [7, 27]. Since **LH** is considered to contain problems which have much lower complexity than hard problems in **P**, the effect of exogenous actions is drastic in complexity terms. Furthermore, problems in **LH** are amenable to parallelization (see [27]).

Under state variables representation (Table 2), the complexity of the problems, with few exceptions increases by an exponential. This increase is intuitively explained by the fact that state variables permit in general an exponentially smaller input representation, which must be unpacked for solving the problem. The exception for constant k in absence of exogenous functions, where the complexity increases from within **LH** to co-**NP**, is intuitively explained by the fact that the quantifier "$\exists a \in \mathcal{A}_{agent} \cap poss(s)$" in equation (1), as opposed to "$\forall s' \in \mathcal{S}$", ranges over a polynomial set of values (in the input size), and thus can be deterministically eliminated.

The **EXP**-completeness means that the problems are provably intractable, i.e., have an exponential lower bound in this setting. Even in the "cheapest" cases under state variable representation, the problems are intractable. Exogenous actions cannot be compiled efficiently away in the same cases as under enumerative representation.

7.2 Enumerative representation

We start with the case of enumerative representation. Our first result is the following.

Theorem 11 Problem k-MAINTAINABILITY is **P**-complete (under logspace reductions). The **P**-hardness holds under the restriction that $k = f(A, S, E)$ is any function of A, S, and E such that $f(A, S, E) \geq 1$ (in particular, for fixed $k \geq 1$), even if in addition all actions are deterministic and there is only one exogenous action.

Proof. The membership of k-MAINTAINABILITY in **P** follows from Corollary 9.

We prove **P**-hardness under the stated restriction by a reduction from deciding logical entailment $\pi \models q$ of a propositional atom q from a propositional Horn logic program (PHLP) π, which is a set of rules of the form

$$b_0 \leftarrow b_1, \ldots, b_n, \qquad n \geq 0, \tag{2}$$

and each b_i is a propositional atom from an underlying atom set At; b_0 is the head and $b_1, \ldots, b_n$ is the body of the rule.

As well-known, $\pi \models q$ holds iff there is a sequence of rules $r_1, r_2, \ldots, r_m$, $m \geq 1$, from π where r_i is of form $b_{i_0} \leftarrow b_{i_1}, \ldots, b_{i_n}$, such that $\{b_{i_1}, \ldots, b_{i_n}\} \subseteq \{b_{1_0}, \ldots, b_{i-1_0}\}$, for all $i \in \{1, \ldots, m\}$ (thus in particular, $1_n = 0$) and $b_{m_0} = q$, called a *proof of q from π*. Informally, q is derived by successive application of the rules $r_1, \ldots, r_m$, where r_i "fires" after all previous rules $r_1, \ldots, r_{i-1}$ have fired.

A natural idea is to represent backward rule application $r_m, r_{m-1}, \ldots, r_1$ through agent actions; for a rule r of form (2), there is an agent action a_r which applied to a state s_{b_0} representing b_0, brings the agent nondeterministically to any state s_{b_i} representing b_i, $i \in \{1, \ldots, n\}$. Given a state s_q encoding q, $S = \{s_q\}$ is maintainable w.r.t. a set of states E encoding the facts in π if q has a proof from π. However, this does not account for the restriction that $k = f(A, S, E)$ for any such f. The key for this is to establish the result for the extremal case where $k = 1$ is constant (i.e., for 1-MAINTAINABILITY) and then to extend it to the general case.

Using a constrained rule format in π and an exogenous action, we can emulate nondeterministic agent actions and sequences of agent actions with some coding tricks by alternating sequences of deterministic

agent and exogenous actions, such that provability of q from π corresponds to 1-maintainability of S w.r.t. a set E in a system A constructible in logarithmic workspace from q and π.

Without loss of generality, we assume that each rule has either zero or two atoms in the body (i.e., $n = 0$ or $n = 2$ in (2)). We construct from π and q a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, sets of states S and E, a set $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a function exo as follows:

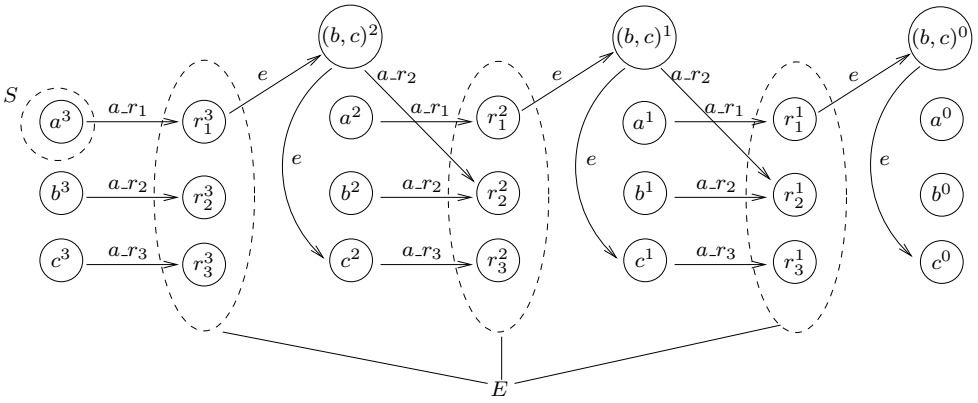

Figure 3: Transition diagram of the system for $\pi = \{a \leftarrow b, c;\ b \leftarrow\ ;\ c \leftarrow\}$ and $q = a$ (S and E encircled).

1. $\mathcal{S}$: For each atom f in π and rule $r \in \pi$, $f^0, \ldots f^m$ and $r^1, \ldots, r^m$ are states in $\mathcal{S}$. Furthermore, if the body of r is u, v then $(u, v)^0, \ldots, (u, v)^{m-1}$ are states in $\mathcal{S}$.
2. $\mathcal{A} = \{a\_r \mid r \in \pi\} \cup \{e\}$.
3. Φ: For any rule $r \in \pi$ with head f, $\Phi(a\_r, f^i) = \{r^i\}$ for $i \in \{1, \ldots, m\}$ and $\Phi(a\_r, (u, v)^i) = \{r^i\}$, for $(u, v)^i \in \mathcal{S}$, $i \in \{1, \ldots, m-1\}$. If moreover r has body u, v, then $\Phi(e, r^i) = \{(u, v)^{i-1}\}$, and $\Phi(e, (u, v)^{i-1}) = \{v^{i-1}\}$, for $i \in \{1, \ldots, m-1\}$. In all other cases, $\Phi(a, s) = \emptyset$.
4. $poss$: For each state s, $poss(s) = \{a \in \mathcal{A} \mid \Phi(a, s) \neq \emptyset\}$.
5. $E = \{r^1, \ldots, r^m \mid r \in \pi\}$
6. $S = \{q^m\}$.
7. $\mathcal{A}_{agent} = \mathcal{A} \setminus \{e\}$.
8. exo: for all rules $r \in \pi$ of form $f \leftarrow u, v$, $exo(r^i) = \{e\}$ for $i \in \{1, \ldots, m\}$ and $exo((u, v)^j) = \{e\}$ for $j \in \{1, \ldots, m-1\}$. For all other states s, $exo(s) = \emptyset$.

The transition diagram for the system constructed for $\pi = \{a \leftarrow b,\ b \leftarrow,\ c \leftarrow\}$ is shown in Figure 7.2. Intuitively, the state f^i encodes that f can be derived from π with a proof of length at most i. This is propagated in backward rule application. Each agent action a_r selects a rule r to prove an atom f; if the rule has a body u, v, the exogenous action pushes the agent to prove both u (from (u, v)) and v within decreased recursion depth.

We claim that $\pi \models q$ iff there exists some 1-maintaining control K for S with respect to E in A.

Suppose first that $\pi \models q$. We then construct a 1-maintaining control K for S with respect to E as follows. Let $P = r_1, \ldots, r_k$ be a proof of q from π such that, without loss of generality, all rules r_i have different heads. Set $D = \{q^m\}$ and iterate the following until D remains unchanged: For each $f^i \in D$ resp. $(u,v)^i \in D$, $i \geq 0$, let r_j be the rule with head f resp. u in P. Define $K(f^i) = \{a\_r_j\}$ resp. $K((u,v)^i) = \{a\_r_j\}$, and add, if r_j has body u', v' the states $(u,v)^{i-1}$ and v'^{i-1} to D. Since P is a proof of q from π, the rule r_j always exists, and for each state s in $\mathit{Closure}(S, A_{K,exo}) \setminus E$ (=D), $K(s)$ is defined and $\Phi(K(s), s)$ yields some state in E. Hence, K is a 1-maintaining control for S with respect to E in A.

Conversely, suppose K is a 1-maintaining control for S with respect to E in A. Without loss of generality, $K(s)$ is undefined for all states $s \in E$. An easy induction on $i \geq 1$ shows that for each $f^i \in \mathit{Closure}(S, A_{K,exo})$ resp. $(u,v)^i \in \mathit{Closure}(S, A_{K,exo})$, it holds that $\pi \models f$ resp. $\pi \models u$ and $\pi \models v$. For i=1, suppose first $K(f^1) = a\_r$. Rule r must have form $f \leftarrow$; otherwise, some states $(u,v)^0$, v^0 would be in $\mathit{Closure}(S, A_{K,exo})$, which contradicts that K is a 1-maintaining control. Hence, $\pi \models f$. Next suppose $K((u,v)^1) = a\_r$. Then, for similar reasons, r must be of form $u \leftarrow$, hence $\pi \models u$. Furthermore, $v^1 \in \mathit{Closure}(S, A_{,exo})$ and as already established $\pi \models v$. For $i > 1$, suppose $K(f^i) = a\_r$. Then either r is of form $f \leftarrow$ and thus $\pi \models f$, or of form $f \leftarrow u, v$. In the latter case, $(u,v)^{i-1} \in \mathit{Closure}(S, A_{K,exo})$ and hence, by the induction hypothesis, $\pi \models u$ and $\pi \models v$. Consequently, $\pi \models f$. Similarly, if $K((u,v)^i) = a\_r$, then either r is of form $u \leftarrow$ or of form $u \leftarrow u', v'$ and $(u',v')^{i-1} \in \mathit{Closure}(S, A_{K,exo})$, which by the induction hypothesis implies $\pi \models u'$ and $\pi \models v'$, thus $\pi \models u$. Since $v^i \in \mathit{Closure}(S, A_{K,exo})$, as already established $\pi \models v$. Consequently, $\pi \models f$. This proves the statement for $i > 1$, and concludes the induction. Since $q^m \in \mathit{Closure}(S, A_{K,exo})$, we have $\pi \models q$. This proves our claim.

Notice that A, S and E can be constructed in logarithmic workspace from π and q. This proves **P**-hardness of 1-MAINTAINTABILITY. An easy observation is that every agent action in the system A leads to some state in the set E described. Hence, S is 1-maintainable with respect to E in A iff S is k-maintainable with respect to E in A for any $f(A,S,E)$ such that $f(A,S,E) \geq 1$. Hence, **P**-hardness under the stated restriction follows. □

The following result is immediate from this result and the fact that maintainability is equivalent to k-maintainability where $k = |\mathcal{S}|$ is the number of states.

Corollary 12 ω-MAINTAINABILITY is **P**-complete. The **P**-hardness holds even if all actions are deterministic and there is only one exogenous action.

The following result states a further **P**-complete restriction of the above problems.

Theorem 13 k-MAINTAINABILITY and ω-MAINTAINABILITY without exogenous actions are **P**-complete.

Proof. Membership in **P** was established above. The **P**-hardness follows from Theorem 11 by merging the (single) exogenous action e into the agent actions as follows: For each state s such that $e \in exo(s)$, redefine every action $a \in poss(s) \cap \mathcal{A}_{agent}$ by $\Phi(s,a) := \Phi(s,a) \cup \Phi(s,e)$. It is easy to see that given S and E, S is $|\mathcal{S}|$-maintainable w.r.t. E in the resulting system A' iff S is $|\mathcal{S}|$-maintainable w.r.t. E in A. Furthermore, A' is computable in logspace from A. This implies the result. □

The hardness results above are at the border of the hardness frontier, in the sense that in the absence of exogenous actions and, in case of ω-MAINTAINABILITY also nondeterminism, the problems are no longer **P**-hard. The following lemma gives a useful characterization of k-maintainability for this purpose.

Lemma 14 Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{agent} \subseteq \mathcal{A}$, and a set of states E, a set of states S is k-maintainable with respect to E in absence of exogenous actions (i.e., exo is void), $k \geq 0$, iff $r_k(s)$ as in (1) holds for all $s \in S$.

Proof. For the only if direction, consider any 1-maintaining control K which without loss of generality is undefined on every $s \in E$. For every state $s \in \mathit{Closure}(S, A_{K,exo}) = \mathit{Closure}(S, A_K)$, let d_s be the distance of s from E under K, i.e., the largest i such that $\sigma = s_0, s_1, \ldots, s_i \in \mathit{Unfold}_k(s, A, K)$ where $s_0 = s$. By an easy induction on $d_s \geq 0$, we obtain using $K(s)$ as witness for a in (1), that $r_{d_s}(s), r_{d_s+1}(s), \ldots, r_k(s)$ must hold for s. Hence, $r_k(s)$ holds for every $s \in S$.

Conversely, let for each $s \in \mathcal{S}$ be i_s the least integer i such that $r_i(s)$ holds. If $i_s > 0$, then define $K(s) := a$ for some arbitrary action $a \in \mathcal{A}_{agent} \cap poss(s)$ witnessing (1) for $i + 1 = i_s$, otherwise (i.e., if $i_s = 0$ or $r_i(s)$ does not hold for any $i \geq 0$) let $K(s)$ undefined. Then, K is a k-maintaining control for S with respect to E, since by definition of the relations r_i, for each $s \in \mathit{Closure}(S, A_K)$, and $\sigma = s_0, s_1, \ldots, s_l \in \mathit{Unfold}_k(s, A, K)$ such that $s_0 = s$ it holds that $l \leq k$ and $s_l \in E$ (recall that, as tacitly assumed, $\Phi(a, s) \neq \emptyset$ for each $a \in poss(a)$). Hence, S is k-maintainable with respect to E. □

We then establish the following result.

Theorem 15 k-MAINTAINABILITY and ω-MAINTAINABILITY for systems with only deterministic actions and no exogenous actions are **NL**-complete.

Proof. In this case, deciding $r_i(s)$ for given $s \in \mathcal{S}$ and $i \geq 0$ is in **NL**: If $s \notin E$, a proper a in (1) and $s' = \Phi(s, a)$ can be guessed and, recursively, $r_{k-1}(s')$ established, maintaining a counter i. This is feasible in logarithmic workspace in the representation size of A. By looping through all $s \in \mathcal{S}$, it thus follows from Lemma 14 that deciding whether S is k-maintainable with respect to E, where $k \leq |\mathcal{S}|$, is nondeterministically feasible in logarithmic workspace. This implies **NL**-membership of k-MAINTAINABILITY and ω-MAINTAINABILITY. The hardness follows from a simple reduction of the well-known **NL**-complete REACHABILITY problem [44] to k- resp. ω-MAINTAINABILITY: Given a directed graph $G = (V, E)$ and nodes $s, t \in V$, decide whether there is a directed path from s to t in G. Define $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ such that $\mathcal{S} = \mathcal{A} = V$, $\Phi(v, w) = w$, and $poss(v) = \{w \mid v \to w \in E\}$. Then, for $\mathcal{A}_{agent} = \mathcal{A}$, $S = \{s\}$ is $|V|$-maintainable w.r.t. $E = \{t\}$ in A iff there is a directed path from s to t in G. Clearly, A is constructible in logarithmic workspace from G. This shows the **NL**-hardness.□

In case of constant k, equation (1) is decidable by a straightforward deterministic recursive procedure in logarithmic workspace, even under nondeterminism, since the recursion depth is bounded by a constant and each recursion level requires only logarithmic work space. Hence, k-MAINTAINABILITY is decidable in logarithmic space. A finer grained analysis that it is within the class $\Pi^{\log}_{k+1}$ of the logarithmic time hierarchy, which is a much better upper bound and makes completeness for logspace (under suitable reductions) fairly unlikely.

We assume that the input I of k-MAINTAINABILITY for fixed k, is a relational structure $\mathcal{M}_I$ with universe $U(\mathcal{M}_I) = \mathcal{S} \cup \mathcal{A}$, and relations over $U(\mathcal{M}_I)$ for the predicates $in\_Phi(s, a, s')$, $in\_poss(s, a)$, $in\_exo(s, a)$, $in\_S(s)$ and $in\_E(s)$ from above, and relations for the additional predicates $ag\_act(a)$, $in\_\mathcal{S}(s)$, and $in\_\mathcal{A}(a)$ representing membership $a \in \mathcal{A}_{agent}$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$ for each $s, a \in U(_M)$, respectively. The structure $\mathcal{M}_I$ is encoded in a standard way by a bit-string [27].

Theorem 16 Problem k-MAINTAINABILITY for systems without exogenous actions is in $\Pi^{\log}_{2k+1}$ (=co-$\Sigma^{\log}_{2k+1}$), if $k \geq 0$ is constant.

Proof. Any first-order formula $\psi_1 \vee Qx\,\psi_2$ resp. $\psi_1 \wedge Qx\,\psi_2$ such that ψ_1 has no free variables and $Q \in \{\exists, \forall\}$, is logically equivalent to $Qx(\psi_1 \vee \psi_2)$ resp. $Qx(\psi_1 \wedge \psi_2)$. Exploiting this, $r_k(s)$ in (1) can be written, using the vocabulary from above, as a first-order formula $\phi_k(x)$ in prenex form $\exists x_1 \forall x_2 \exists x_3 \cdots Q_k x_k \psi(x_1, \ldots, x_k, x)$ where $\psi(x_1, \ldots, x_k, x)$ is quantifier-free, such that for any element $s \in U(\mathcal{M}_I)$ of an input structure $\mathcal{M}$, the sentence $in\_\mathcal{S}(s) \wedge \phi_k(s)$ is true on $\mathcal{M}$ iff $r_k(s)$ holds. Hence, by Lemma 14, k-maintainability of S w.r.t. E in A is definable by a Π_{k+1} prenex sentence $\forall x_0 \exists x_1 \cdots Q_k x_k \psi'(x_0, x_1, \ldots, x_k)$, where $\psi'(x_0, x_1, \ldots, x_k)$ is quantifier-free, on the above vocabulary. Whether a fixed such sentence is false on a given structure $\mathcal{M}_I$ can be decided by an alternating Turing machine, starting in an existential state, in logarithmic time using k alternations [7, 27]. Hence, the problem is in co-$\Sigma^{\log}_{k+1}$ = $\Pi^{\log}_{2k+1}$. □

We remark that the hardness results in this section can be further strengthened to the case where only 2 agent actions are available, but leave a proof of this to the interested reader.

7.3 State variables

The following is an easy lemma, which in combination with the results in the previous subsection implies most upper bounds in Table 2.

Lemma 17 For any instance of k-MAINTAINABILITY resp., ω-MAINTAINABILITY in which states are represented by variables, the corresponding instance in ordinary (enumerative) form can be generated in polynomial workspace.

Using this lemma, we then prove the following result.

Theorem 18 Under state representation by variables, k-MAINTAINABILITY is **EXP**-complete. The **EXP**-hardness holds under the restriction that $k = f(A, S, E)$ is any function of A, S, and E such that $f(A, S, E) \geq 1$ (in particular, for fixed $k \geq 1$), even if in addition all actions are deterministic and there is only one exogenous action.

Proof. Membership in **EXP** follows easily from Lemma 17 and Theorem 11. The **EXP**-hardness is shown by a reduction from deciding inference $\pi \models p(t)$ of a ground atom $p(\bar{c})$ from a function-free Horn logic program π with variables (i.e., a datalog program), which consists of rules of the form

$$p_0(\bar{t}_0) \leftarrow p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n), \qquad n \geq 0, \tag{3}$$

where each p_i is the name of a predicate of arity $a_i \geq 0$ and $\bar{t}_i = t_{i,1}, \ldots, t_{i,n}$ is a list of constants and variables $t_{i,j}$; $p_0(\bar{t}_0)$ is the head and $p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n)$ the body of the rule.

It holds that $\pi \models p(\bar{c})$ iff there is a sequence rules r_i of the form $p_{i_0}(\bar{t}_{i_0}) \leftarrow p_{i_1}(\bar{t}_{i_1}), \ldots, p_{i_n}(\bar{t}_{i_n})$ and substitutions θ_i for r_i, i.e., a mappings from the variables in r_i to the set of constants C_π in π, such that $\{p_{i_1}(\bar{t}_{i_1}\theta_i), \ldots, p_{i_n}(\bar{t}_{i_n}\theta_i)\} \subseteq \{p_{1_0}(\bar{t}_{1_0}\theta_1), \ldots, p_{i-1_0}(\bar{t}_{i-1_0}\theta_{i-1})\}$, for all $i \in \{1, \ldots, m\}$ (thus in particular, $1_n = 0$) and $p_{m_0}(\bar{t}_{m_0}\theta_m) = p(\bar{c})$, called a *proof of* $p(\bar{c})$ *from* π. Informally, $p(\bar{c})$ is derived by successive application of the rule instances $r_1\theta_1, \ldots, r_m\theta_m$, like in a propositional logic program.

Deciding whether $\pi \models p(t)$ is well-known to be **EXP**-complete, cf. [13]. The construction is similar in spirit to the one in proof of Theorem 11 but more involved.

To prove **EXP**-hardness of k-MAINTAINABILITY under the given restriction, we first focus on 1-MAINTAINABILITY, and we describe how to reduce $\pi \models p(\bar{c})$ in logarithmic workspace to deciding 1-maintainability of a set of states S w.r.t. a set of states E in an agent system A.

Without loss of generality, we make the following assumptions on π and $p(\bar{c})$:

- The set of constants occurring in π, C_π, is $\{0, 1\}$;
- each rule r in π has either zero or two atoms in the body;
- all rules in r are safe, i.e., each variable X occurring in the head of a rule r also occurs in the body;
- π uses only one predicate, p;
- $c = (0, 0, \ldots, 0)$.

Any problem $\pi \models p(\bar{c})$ can be transformed to an equivalent one of this form in logarithmic workspace.

Similar as in the propositional case, the idea is to represent a reversed proof $r_m, \theta_m, \ldots, r_1\theta_1$ of $p(\bar{c})$ from π through agent actions, and model backward rule applications through agent actions; note that m ranges from 1 to 2^{a_p}, where a_p is the arity of p (thus m requires a_p bits). The problem here which makes this more complex is the fact that we must, for each rule r_i, also take θ_i into account. If r_i has a nonempty body, the candidates for θ_i are systematically generated by alternating agent and exogenous actions. For each possible such θ_i, the derivation of the body atoms $p(\bar{t}_{i_2}\theta_i)$ and $p(\bar{t}_{i_2}\theta_i)$ is then explored.

More precisely, for each ground atom $p(\bar{c})$, and $m \in \{0, \ldots, 2^{p_a}\}$, we have a state $(\bar{c}, m, prove)$ outside E which intuitively says that $p(\bar{c})$ is derivable within m ($0 \leq m \leq 2^{p_a}$) steps. For each rule r in π, there is an agent action a_r, which is possible on $(\bar{c}, m, prove)$ if $m > 0$ and $p(\bar{c})$ unifies with the head $p(\bar{t})$ of r, and it results in the state $(\bar{c}, m, r, apply)$, which is in E. For r of form $p(\bar{t}) \leftarrow p(\bar{t}_1), p(\bar{t}_2)$, two phases are now established: (1) the selection of a substitution θ for the variables X in r, and (2) the generation of states $(\bar{c}_1, m-1, prove)$ and $(\bar{c}_1, m-1, prove)$, where $\bar{c}_1 = \theta_1$ and $\bar{c}_2 = \theta_2$, for the recursive test.

As for 1) an exogenous action e pushes the agent from $(\bar{c}, m, r, apply)$ to a state $(\bar{c}, m, (0, 0, ..., 0), r, sel\_\theta)$. Here $(0, 0, \ldots, 0)$ is the substitution $\theta : X_1 = 0, \ldots, X_k = 0$ to all variables in r. By executing an agent action inc_θ on this state, this vector is incremented to $(0, 0, ..., 0, 1)$, resulting in a state $(\bar{c}, m, (0, 0, ...0, 1), r, inc_\theta)$ in E, from which e pushes the agent to a state $(\bar{c}, m, (0, 0, ..., 1), r, sel\_\theta)$, where $X_n = 1$ in θ. Here again inc_θ is possible, leading to a state $(\bar{c}, m, (0, 0, ..., 1, 0), r, inc_\theta)$ in E from which e pushes the agent to the state $(m-1, t, (0, 0, ...1, 0), r, sel\_\theta)$. Here again an inc action is possible for the agent etc.

In each state $(\bar{c}, m, \theta, r, sel\_\theta)$ such that $p(t\theta) = \bar{c}$, the agent might alternatively take the action $choose$, which brings her to the state $(\bar{c}, m, \theta, r, chosen_\theta)$ in E, which closes phase 1. The exogenous action e pushes the agent from this state to the state $(m, \bar{t}_1\theta, \bar{t}_2\theta, do\_split)$ out of E. From this state, e pushes the agent further to the state $(\bar{t}_1\theta, m-1, prove)$, and the agent must take at $(m, \bar{t}_1\theta, \bar{t}_2\theta, do\_split)$ the action $split$, which brings her to the state $(\bar{t}_2\theta, m-1, goto\_prove)$ in E, from which e pushes the agent to $(\bar{t}_2\theta, m-1, prove)$. Figure 4 gives a summary of the steps in graphical form.

In this way, the derivation of $p(0, 0, \ldots, 0)$ from π is encoded to deciding 1-maintainability of $S = \{(2^d, (0, 0, ..., 0), prove)\}$ with respect to the set of states E described above. Note that to prove $p(\bar{c})$ from π via rule r, only one instance of $r\theta$ must be chosen; the 1-maintaining control has to single out this θ, by proper placement of the action $chosen_\theta$. The proof of correctness is along the lines of the respective one in Theorem 11.

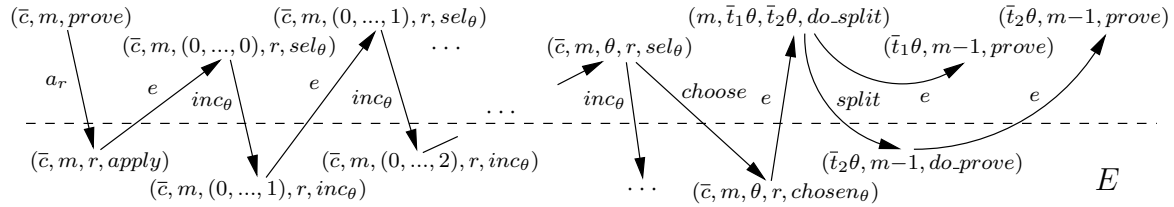

Figure 4: Schematic transition diagram for backward application of rule $r : p(\bar{t}) \leftarrow p(\bar{t}_1), p(\bar{t}_2)$ with substitution θ to prove $p(\bar{c})$.

Given the regular structure of the states and the easy checks and manipulations that need to be done for determining applicability of actions and determining the successor state, respectively, it is not difficult to see that a representation of the above 1-MAINTAINABILITY instance using state variables can be compiled from π and $p(0, 0, \ldots, 0)$ in logarithmic work space (in particular, that the polynomial-time procedures for deciding the membership predicates $in\_Phi(s, a, s')$, $in\_poss(s, a)$, $in\_exo(s, a)$ $in\_S(s)$, and $in\_E(s)$ can be provided in polynomial time). Note that this instance employs only deterministic actions, and there is a single exogenous action. This establishes **EXP**-hardness for 1-MAINTAINABILITY.

Furthermore, for A and E as constructed, each agent action results in a state in E. Thus, k-maintainability of S w.r.t. E in A, for any $k = f(A, S, E)$ such that $f(A, S, E) \geq 1$, is equivalent to 1-maintainability of S w.r.t. E in A. Hence, the reduction shows **EXP**-hardness of $k-$MAINTAINABILITY under the stated restriction. □

Corollary 19 Under state representation by variables, ω-MAINTAINABILITY is **EXP**-complete. The **EXP**-hardness holds even if all actions are deterministic and there is only one exogenous action.

Using Theorem 18 instead of Theorem 11, we can prove the following result similarly as Theorem 13:

Theorem 20 Under state representation by variables and in absence of exogenous actions, k-MAINTAINABILITY and ω-MAINTAINABILITY are **EXP**-complete.

For the case without exogenous actions and with only deterministic actions, we have lower complexity:

Theorem 21 Under state representation by variables, k-MAINTAINABILITY and ω-MAINTAINABILITY for systems with only deterministic actions and no exogenous actions are **PSPACE**-complete.

Proof. By well-known standard methods, a computation composed of a **PSPACE** computation A piped into an **NL** computation B (which is **NPSPACE** in the size of the input for A) can be redesigned as an **NPSPACE** computation. Since **NPSPACE** = **PSPACE**, membership of the problems in **PSPACE** thus follows from Lemma 17 and Theorem 15.

The **PSPACE**-hardness can be shown e.g. by a straightforward reduction from propositional STRIPS planning [9]. Rather than to introduce STRIPS here, we give for completeness sake a simple reduction from SUCCINCT REACHABILITY [44], which is the version of REACHABILITY where $G = (V, E)$ is such that the nodes v are given by the binary vectors $v = (v_1, \ldots, v_n)$, $n \geq 1$, on $\{0, 1\}$ and the problem input consists of a Boolean circuit C_G with $2n$ inputs $v_1, \ldots, v_n$, $w_1, \ldots, w_n$ which outputs

true iff $v \rightarrow w \in E$, and $s = (0, 0, \ldots, 0)$ and $t = (1, 1, \ldots, 1)$. We construct from this an instance of k-MAINTAINABILITY resp. ω-MAINTAINABILITY as follows: $\mathcal{S} = V \times V$, described by $2n$ binary variables $f_1, \ldots, f_{2n}$; $\mathcal{A} = \{inc, arc\} = \mathcal{A}_{agent}$; $\Phi(v \times w, inc) = v \times w'$ such that $w' = w + 1$ modulo 2^n, and $\Phi(v \times w, arc) = w \times (0, 0, \ldots, 0)$ if $v \rightarrow w$ in G and $\Phi(v \times w, arc) = v \times w$ otherwise; $poss(s) = \mathcal{A}$, for each state s. Then, the state $s = (1, 1, \ldots,) \times (0, 0, \ldots,)$ is $|\mathcal{S}|$-maintainable with respect to $E = \{(1, 1, \ldots, 1) \times (1, 1, \ldots, 1)\}$ in A iff $(1, 1, \ldots, 1)$ is reachable from $(0, 0, \ldots, 0)$ in G. A state variable representation of A can be easily generated from the circuit C_G in logarithmic workspace. This implies **PSPACE**-hardness of the problems. □

If the maintenance window is bounded by a constant, the problem is easier.

Theorem 22 Under state representation by variables, k-MAINTAINABILITY for systems without exogenous actions and constant $k \geq 0$ is co-**NP**-complete.

Proof. For a given $s \in \mathcal{S}$, falsity of $r_k(s)$ can be proved by exhibiting (assuming $s \notin E$), for each $a \in \mathcal{A}_{agent} \cap poss(s)$ a witness $w(s, a) \in \mathcal{S}$ such that $w(s, a) \in \Phi(s, a)$ and $r_{k-1}(w(s, a))$ is false, which in recursion can be proved similarly. For constant k, this leads to $O(|\mathcal{A}_{agent}|^k)$ many guesses $w(s, a)$, which is polynomial in the size of the input. By Lemma 14, it thus follows that deciding the complement of k-MAINTAINABILITY is in **NP**. This proves membership in co-**NP**.

The co-**NP**-hardness, for every $k \geq 0$, is a simple consequence that under representation by state variables, deciding whether $S \subseteq E$ is co-**NP**-complete (this can be shown, e.g., by a simple reduction from propositional unsatisfiability). □

8 Discussion and Conclusion

In this paper, we gave a formal characterization of maintenance goals and distinguished it from the notions of stabilizability and temporal goals of the form $\Box\Diamond f$ (over all valid trajectories). 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 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 then gave polynomial time algorithms to compute k-maintainable controls, which are linear-time for small k, and we analyzed the complexity of determining k-maintainability under various assumptions. One interesting aspect of our polynomial time algorithm is the approach that led to its finding: use of SAT encoding, and complexity results regarding the special Horn sub-class of propositional logic.

8.1 Other related work

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 [42], 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 [14]. 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 [2], where an agent executes a cycle of $observe; assimilate; (re)plan\_from\_current\_situation; execute\_part\_of\_the\_plan$.

In other related work, Jensen et al. [28, 29] consider the somewhat dual problem of developing policies that achieve a given goal while there are interferences from the environment. In their model, environment actions and actions of multiple agents are combined to a joint action, by which the system is transferred from the current state to one out of a set of possible successor states. With such nondeterministic transitions, Jensen et al. aim at modeling both an adversial environment and infrequent errors which make an otherwise deterministic action non-deterministic. In [28], they consider constructing policies coping with arbitrarily many interferences of the environment (but without action failure) by an extension of OBDD-based universal planning, and in [29] they consider generating policies which tolerate up to a given number n of errors modeled as "secondary action effects" (caused by improper action execution or environment interference), by reducing it to a so called strong planning problem, which is solved using OBDD based methods. For arbitrarily many environment interferences as in [28], the problem is basically very similar to our problem of unbounded maintainability, but interference in goal states has different significance and goal achievement is not guaranteed because of possible loops. A formal connection between k-maintainable controls and n-fault tolerant policies, if any, remains open. Intuitively, n-fault tolerant plans are easier to construct, since the number of errors that have occurred can be recorded in plan construction and when the limit n is reached, the problem boils down to an ordinary planning problem. For k-maintaining controls, however, each environment interference (even at a goal state) causes a restart which pushes the agent to a new initial state.

In a series of papers [54, 19, 18], Wooldridge and Dunne have formalized the problem of constructing agent control functions and analyzed its complexity in a rich framework, for various kinds of tasks such as "achievement" tasks (where the agent has to bring about a certain goal condition), "maintenance" tasks (where the agent has to avoid that some goal condition is ever satisfied during execution), and combinations thereof [18]. In their framework, action effects and the selection of the agent action by the control may depend on the history of the execution, and most importantly, exogenous actions resp. an adversary are not taken into account. Under restriction to history-independent state transitions and reactive agents, finding controls for achievement tasks in their framework corresponds to finding maintaining controls with an unbounded window of opportunity in our framework. Theorems 15 and 21 correspond to respective results in the Wooldridge-Dunne framework [18].

In AI planning, the seminal STRIPS approach [23] has been one of the most influential approaches. We briefly recall that in STRIPS, states are modeled as sets of propositional atoms and actions as operators which, given that a precondition in terms of a conjunction of literals is true on the current state, transform it to a successor state by removing atoms from a delete list and adding atoms from an add list. A plan for achieving a goal, described by a conjunction of atoms γ, from an initial state S_0 is a sequence of operators $op_1, \ldots, p_n$ which takes the agent from S_0 to a state where γ holds. STRIPS planning has been generalized in several directions, such as conditional effects, nondeterministic actions, or planning under incomplete information and partial observability using conditional and conformant plans, respectively, and a number of papers has considered the computation and complexity of planning in such settings, e.g., [9, 3, 11, 22, 49].

However, like in the framework of Wooldridge and Dunne, in none of these works agent actions and exogenous actions are viewed separately, and thus they are best compared to our framework in absence of exogenous functions. Furthermore, plans per se are conceived as *action strategies* (cf. [49]) in which, in principle, different actions might be taken by the agent if during plan execution the same state is entered

again; however, such looping is a priori excluded if the goal must be achieved under all contingencies. Cimatti et al. [11] consider constructing universal plans akin to our policies, with different semantics for goal achievement, based on OBDD methods and algorithms. In particular, in absence of exogenous actions our maintaining controls correspond to what they call strong solutions for a planning problem. Jensen et al. [28, 29] have generalized this by adversial actions (see above).

As for complexity, Theorem 21, corresponds to the classical result of Bylander [9] that deciding plan existence in propositional STRIPS is **PSPACE**-complete, while Theorem 20 corresponds to Littman's result that conditional planning for STRIPS with nondeterministic actions is **EXPTIME**-complete [34, 49]. In conditional planning, via conditions on the current state branching to subplans is possible, such that an appropriate plan is followed depending on the state evolution. Branching might be modeled by actions and the conditional planning problem, with loops disregarded, as the problem of constructing a maintaining control.

Outside of AI, our notion of k-maintenance is very closely related to the notion of self-stabilization in [15] which is used in characterizing fault-tolerant systems. There the concern is about proving correctness of (hand developed) self-stabilization protocols and achieving self-stabilization for various distributed algorithms such as mutual exclusion. Our algorithm here can be thought of as an algorithm that automatically generates a self-stabilization protocol. Although, this is a new dimension to the existing work on self-stabilization, further research is needed to compare assumptions made in our formulation and the ones in the self-stabilization literature, and overcome them. In particular, often in the self-stabilization literature the global states are composed of local states of various distributed elements and a particular element does not have the access to the complete global state. In those cases one can not directly use the kind of global policies generated by the algorithm in this paper.

8.2 Future work and open issues

There are several directions for further research extending the work of this paper. One direction concerns variations of the maintenance problem, for instance by taking action duration into account. In such scenario, the maintenance goal may be formulated as requirement that the agent reaches some desired state always within a given time frame, if she is not disturbed by the environment. Preliminary investigations suggest that the results in this paper can be extended to handle this setting.

The intractability results for the problems under state variable representations challenges methods and techniques for handling the problem in practice. Suitable heuristics may therefore be researched that allow to solve the problems in many cases in polynomial time, and, in a refined complexity analysis, meaningful tractable cases should be singled out. Furthermore, the issue of computing optimal k-maintenance controls efficiently, in the sense that k is as small as possible (which is trivially polynomially solvable in the enumerative setting), is an interesting issue for variable state representation.

Another issue concerns investigating computational transformations between maintenance and planning. By the complexity results in [34] and this paper, transformations between k-MAINTAINABILITY and conditional planning are feasible in polynomial time. It would be interesting to study different transformations, and to assess possible benefits of these transformations for solving k-MAINTAINABILITY and planning by cross-utilizing different algorithms and implementations (e.g. [11] for planning in nondeterministic domains). In particular a transformation similar to the one in the proof of Theorem 13, with an additional parameter that keeps count the number of agent's actions since the last exogenous action, can[4] be used to compile out exogenous actions and transform finding k-maintainable policies to finding

[4]This transformation increases the number of states by k times. It is unknown if there exist a transformation that can

strong cyclic plans [11]; on the other hand, encodings similar to the one in Section 5.2 for obtaining strong cyclic plans through linear-time Horn logic programming might be interesting.

Acknowledgment This work was partially supported by FWF (Austrian Science Funds) projects P-16536-N04 and Z29-N04, the European Commission under grant IST 2001-37004 WASP, the NSF (National Science Foundation of USA) grant numbers 0070463, and 0412000, NASA grant number NCC2-1232, and an ARDA contract. We would like to acknowledge W. Cushing for his feedback on an earlier draft and S. Gupta and M. Gouda for their clarifications on self-stabilization. Furthermore, we acknowledge comments by J. Rintanen on the ICAPS'04 paper and are grateful for his pointers to related work.

References

[1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

[2] C. Baral, M. Gelfond, and A. Provetti. Representing actions: Laws, observations, and hypothesis. *Journal of Logic Programming*, 31:201–243, 1997.

[3] C. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122(1-2):241–267, 2000.

[4] C. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning with temporal goals. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 509–514. Morgan Kaufmann, 2001.

[5] C. Baral and T. Son. Relating theories of actions and reactive control. *Electronic Transactions on Artificial Intelligence*, 2(3-4):211–271, 1998.

[6] C. Baral and J. Zhao. Goal specification in presence of non-deterministic actions. In *Proceedings of ECAI'04*, pages 273–277, 2004.

[7] D. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41:274–306, 1990.

[8] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[9] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69:165–204, 1994.

[10] S. Ceri and J. Widom. Deriving production rules for constraint maintenance. In *Proceedings VLDB-90*, pages 566–577, 1990.

[11] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2):35–84, 2003.

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

[13] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[14] G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Proc. Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 453–465, 1998.

eliminate exogenous actions without increasing the number of states, and yet is able to model the notion of k-maintainability.

[15] E. Dijkstra. A theory of the learnable. *Commun. ACM*, 17(11):643–644, 1974.

[16] W. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn theories. *Journal of Logic Programming*, 3:267–284, 1984.

[17] M. Drummond. Situation control rules. In *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, pages 103–113, 1989.

[18] P. Dunne, M. Laurence, and M. Wooldridge. Complexity results for agent design problems. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):19–36, 2003.

[19] P. Dunne and M. Wooldridge. , atal 2000, boston, ma, usa, july 7-9, 2000, proceedings. In C. Castelfranchi and Y. Lespérance, editors, *Proceedings 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages (ATAL)*, volume 1986 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2001.

[20] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative problem-solving using the DLV system. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, 2000.

[21] E. Emerson. Temporal and modal logics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16. Elsevier Science Publishers B.V. (North-Holland), 1990.

[22] K. Erol, V. Subrahmanian, and D. Nau. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76:75–88, 1995.

[23] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[24] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[25] M. Gelfond and V. Lifschitz. Representing action in extended logic programs. In *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP'92)*, pages 559–573. MIT Press, 1992.

[26] M. L. Ginsberg. Universal planning: An (almost) universally bad idea. *AI Magazine*, 10(4):40–44, 1989.

[27] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[28] R. M. Jensen, M. M. Veloso, and M. H. Bowling. Obdd-based optimistic and strong cyclic adversarial planning. In *Proceedings 6th European Conference on Planning (ECP-01)*, 2001.

[29] R. M. Jensen, M. M. Veloso, and R. E. Bryant. Fault tolerant planning: Toward probabilistic uncertainty models in symbolic non-deterministic planning. In S. Zilberstein, J. Koehler, and S. Koenig, editors, *Proceedings 14th International Conference on Automated Planning and Scheduling (ICAPS 2004), Whistler, British Columbia, Canada, June 3-7, 2004*, pages 335–344, 2004.

[30] F. Kabanza, M. Barbeau, and R. St-Denis. Planning control rules for reactive agents. *Artificial Intelligence*, 95(1):67–113, 1997.

[31] L. P. Kaelbling and S. J. Rosenschein. Action and planning in embedded agents. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 35–48. The MIT Press: Cambridge, MA, USA, 1990.

[32] C. Kuratowski. *Topology I*. Academic Press, New York, 1966.

[33] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 2004. To appear. Available via `http://www.arxiv.org/ps/cs.AI/0211004`.

[34] M. L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings AAAI/IAAI 1997*, pages 748–754, 1997.

[35] P. Maes, editor. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. The MIT Press: Cambridge, MA, USA, 1990.

[36] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems, Specification*. Springer-Verlag, 1992.

[37] M. Minoux. LTUR: a simplified linear time unit resolution for Horn formulae and computer implementation. *Information Processing Letters*, 29:1–12, 1988.

[38] M. Nakamura and C. Baral. Invariance, maintenance and other declarative objectives of triggers – a formal characterization of active databases. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, editors, *Proceedings First International Conference on Computational Logic - CL 2000*, number 1861 in LNAI, pages 1210–1224. Springer Verlag, July 2000.

[39] M. Nakamura, C. Baral, and M. Bjæreland. Maintainability: a weaker stabilizability like notion for high level control. In *Proceedings National Conference on AI (AAAI '00), July 30-August 3, 2000, Austin, Texas*, pages 62–67. AAAI Press, 2000.

[40] I. Niemelä, P. Simons, and T. Syrjänen. Smodels: A system for answer set programming. In C. Baral and M. Truszczyński, editors, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'2000)*, Breckenridge, Colorado, USA, April 2000.

[41] R. Niyogi and S. Sarkar. Logical specification of goals. In *Proceedings 3rd International Conference on Information Technology*, pages 77–82. Tata McGraw-Hill, July 2000.

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

[43] O. Ozveren, A. Willsky, and P. Antsaklis. Stability and stabilizability of discrete event dynamic systems. *J. ACM*, 38(3):7300–752, 1991.

[44] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[45] K. Passino and K. Burgess. *Stability Analysis of Discrete Event Systems*. John Wiley and Sons, 1998.

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

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

[48] R. Reiter. *Knowledge in Action: Logical Foundation for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

[49] J. Rintanen. Complexity of planning with partial observability. In S. Zilberstein, J. Koehler, and S. Koenig, editors, *Proceedings 14th International Conference on Automated Planning and Scheduling (ICAPS 2004), Whistler, British Columbia, Canada, June 3-7, 2004*, pages 345–354, 2004.

[50] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, June 2002.

[51] E. Sontag. Stability and stabilization: Discontinuities and the effect of disturbances. In F. Clarke and R. Stern, editors, *Proceedings NATO Advanced Study Institute*, pages 551–598. Kluwer, July 1998.

[52] D. Weld and O. Etzioni. The first law of robotics (a call to arms). In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1042–1047. AAAI Press, 1994.

[53] J. Widom and S. Ceri, editors. *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, 1996.

[54] M. Wooldridge. The computational complexity of agent design problems. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*. IEEE Press, 2000.