A Polynomial-time Algorithm for Constructing k-Maintainable Policies*

Chitta Baral
Department of Computer Science and Engineering
Arizona State University Tempe, Arizona 85287
chitta@asu.edu

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

Abstract

In this paper we present a polynomial time algorithm for constructing k-maintainable policies (Nakamura, Baral, & Bjareland 2000). Our algorithm, in polynomial time, constructs a k-maintainable control policy, if one exists, or tells that no such policy 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. We then give a logic programming implementation of our algorithm and use it to give a standard procedural algorithm. We then present several complexity results about constructing k-maintainable controls, under different assumptions such as $k = 1$, and compact representation.

Introduction and Motivation

Consider an agent who is assigned the goal of ‘maintaining a room clean’. There are various possible interpretation of this goal. A strict interpretation would be that the room should be always clean. This can be expressed in linear temporal logic as $\Box\, clean$. A less stricter interpretation of it would be to allow the room to get unclean (say while it is being used) but with a guarantee that it will be eventually clean. This can be expressed in linear temporal logic as $\Box\, \Diamond\, clean$. There are two issues with this representation. First, it does not give a bound on how soon $clean$ should be true after $\neg clean$ becomes true. For the second issue, consider the case when the agent is not allowed to clean the room while it is being used and the room is being continuously used. In that case we can not blame the agent for the status of the room. But we can seek a different kind of guarantee. We can demand that the agent give a guarantee that as long as it is not interfered with (i.e., is allowed to clean) for k steps (or k units of time) it will have the room clean after that. This is formulated as k-maintainability in (Nakamura, Baral, & Bjareland 2000). When k is finite it is referred to simply as “maintainability”. This notion was earlier discovered (Dijkstra 1974) in the context of distributed systems where it was referred to as self-stabilization.

Another example in support of the intuition behind maintainability is the notion of maintaining the consistency of a database (Ceri & Widom 1990). 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 of service attacks. In this case we can not fault the triggers saying that they do not maintain the consistency of the database. They do. It is just that they need to be given a *window of opportunity* or a respite from continuous harassment from the environment to bring about the additional changes necessary to restore database consistency.

Another example is a mobile robot (Brooks 1986; Maes 1991) 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 capturing our intuition about ‘maintaining f’ is because $\Box\Diamond f$ is defined on trajectories which do not distinguish between transitions due to agent 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 not interfered by the environment during them; and

(ii) where the agent really does not even try.

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

*This work was partially supported by FWF (Austrian Science Funds) projects P-16536-N04 and Z29-N04, NSF (National Science Foundation of USA) grant number 0070463 and NASA grant number NCC2-1232. The major part of this work was done when Chitta was visiting Vienna University of Technology during May 2003.

1. 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 proceed to give a logic programming implementation of this method, and finally distill from it a standard procedural algorithm.
2. 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.

Background: Actions, states, control policies and k-maintainability

In this section we present a slightly revised versions of the definitions for k-maintainability presented in (Nakamura, Baral, & Bjareland 2000). We start with defining the notion of a system that is used in the discrete event dynamic systems community (Ozveren, Willsky, & Antsaklis 1991).

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, $\mathcal{A}_{ag}$, 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.

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 (Gelfond & Lifschitz 1992; Fikes & Nilson 1971). 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 control 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 case, $\Phi(s, a)$ is *deterministic*, i.e., $\Phi(s, a)$ is a singleton if nonempty.

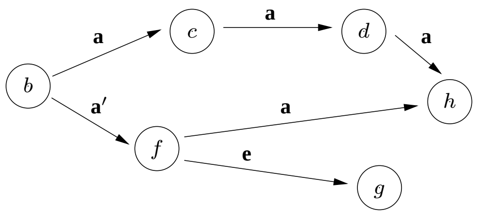

Figure 1: Transition diagram of system A

We now formally define the notion of a control and a super-control policy.

Definition 2 (Control and super-control policy) Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ and a set $\mathcal{A}_{ag} \subseteq \mathcal{A}$ of agent actions,

- a *control policy for* A *w.r.t.* $\mathcal{A}_{ag}$ is a partial function $K : \mathcal{S} \rightarrow \mathcal{A}_{ag}$, such that $K(s) \in poss(s)$ whenever $K(s)$ is defined.
- a *super-control policy for* A *w.r.t.* $\mathcal{A}_{ag}$ is a partial function $K : \mathcal{S} \rightarrow 2^{\mathcal{A}_{ag}}$ such that $K(s) \subseteq poss(s)$ and $K(s) \neq \emptyset$ whenever $K(s)$ is defined.

As we mentioned earlier, the 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. The 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 $\mathcal{A}_{ag} \subseteq \mathcal{A}$ 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 K (mapping a relevant part of $\mathcal{S}$ to $\mathcal{A}_{ag}$) 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. *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 (Ginsberg 1989) - of the number of control rules.* We now formally define the notions of closure and maintainability.

Definition 3 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 satisfies the following conditions: (i) $s \in R(A,s)$, and (ii) if $s' \in R(A,s)$, and $a \in poss(s')$, then $\Phi(s', a) \subseteq R(A,s)$. □

Definition 4 (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 $\mathit{Closure}(S, A)$, is defined by $\mathit{Closure}(S, A) = \bigcup_{s \in S} R(A,s)$. □

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

Next we define the notion of unfolding a control.

Definition 5 **($\mathit{Unfold}_k(s, A, K)$)** Let $A{=}(\mathcal{S},\ \mathcal{A},\ \Phi,\ poss)$ be a system, let $s{\in}\mathcal{S}$, and let K be a control for A. Then $\mathit{Unfold}_k(s, A, K)$ 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 $\mathit{Unfold}_k(s, A, K)$ 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 $\mathit{Unfold}_k(s, A, K)$ 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 6 **(k-Maintainability)** Given a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set of agents action $\mathcal{A}_{ag} \subseteq \mathcal{A}$, and a specification of exogenous action occurrence exo, we say that a control[1] K for A w.r.t. $\mathcal{A}_{ag}$ *k-maintains $S \subseteq \mathcal{S}$ with respect to $E \subseteq \mathcal{S}$*, where $k{\geq}0$, if it holds for each state $s \in \mathit{Closure}(S, A_{K,exo})$ and each sequence $\sigma = s_0, s_1, \ldots, s_l$ in $\mathit{Unfold}_k(s, A, K)$ with $s_0 = s$ 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. Furthermore, S (resp. A) is called *maintainable* w.r.t E, if S (resp. A) is *k-maintainable* w.r.t. E for some $k \geq 0$. □

Example 2 Reconsider the system A in Figure 1. Let us assume that $\mathcal{A}_{ag}$ = { **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, $\mathit{Closure}(\{b\}, A_{K,exo}) = \{b,c,d,h\}$ and $\mathit{Unfold}_3(b, A, K) = \{\langle b,c,d,h\rangle\}$, $\mathit{Unfold}_3(c, A, K) = \{\langle c,d,h\rangle\}$, and $\mathit{Unfold}_3(d, A, K) = \{\langle d,h\rangle\}$; 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**$'$. □

Polynomial time methods to construct k-maintainable control policies

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 (Drummond 1989) and reactive control in the literature. But there have been far fewer efforts (Kabanza, Barbeau, & St-Denis 1997) to define correctness of such control rules[2], and to automatically construct correct control rules. In (Kaelbling & Rosenschein 1991), 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 (Baral & Son 1998) 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 (Nakamura, Baral, & Bjareland 2000) 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 (Jensen, Veloso & Bryant 2003) – 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 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 eases 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;

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

[2] 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.

(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 (Eiter *et al.* 2000) or Smodels (Niemelä & Simons 1997) which extend Horn logic programs by nonmonotonic negation. These solvers allow to compute efficiently the least model and some maximal model of a Horn theory, which 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 $\mathcal{A}_{ag} \subseteq \mathcal{A}$, 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., if both functions are given by their graphs (i.e., in a table).

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 $s_0, s_1, \ldots, s_k$. Intuitively, s_i will denote that (i) 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,*" and that (ii) from each state s' reachable from s, there is an a-path from s' to E of length at most k.

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 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, $1 \leq i \leq k$:

$$s_i \Rightarrow \bigvee_{s' \in PS(s)} s'_{i-1}, \quad \text{where}$$

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

(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. For (0), (1), (4), and (5) it is ok to assume that the intuitive meaning of s_i is as given by (i). Thus, 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 finally (5) that for any state s outside E, there is no a-path from s to E of length 0. Ignoring part (ii) of the meaning of s_i, 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. For (2), let us consider the full intuitive meaning of s_i. The 'if' part of (2) expresses that there is an a-path from s to E of length at most k, and for each state s^* reachable from s, there is an a-path from s^* to E of length at most k. The 'then' part of (2) expresses that there is an a-path from s' to E of length at most k, and for each state s'' reachable from s', there is an a-path from s'' to E of length at most k. The link between the s in the 'if' part and the s' in the 'then' part is that an exogenous action takes one from s to s'. Thus (2) follows from the intuitive meaning of s_i.

Given any model M of $sat(I)$, we can extract a desired control K from it by defining $K(s) = a$, if s is ok to be in the closure for k-maintenance (indicated by truth of s_k in M) but outside E, and 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 1 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$ where Φ is deterministic, a set $\mathcal{A}_{ag} \subseteq \mathcal{A}$, 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., $s_0, s_1, \ldots, s_{j^*-1}$ 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 $K^+_M : \mathcal{S} \rightarrow 2^{\mathcal{A}_{ag}}$ defined on $C_M \setminus E$ such that

$$\begin{aligned} K^+_M(s) \quad = \quad & \{a \in \mathcal{A}_{ag} \cap poss(s) \mid \Phi(s, a) = s', \\ & s' \in C_M, \ell_M(s') < \ell_M(s)\}, \end{aligned}$$

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

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

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 $s \in E \cap S$:

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

(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

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

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

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

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

$\overline{s_0}$.

Here, $\perp$ denotes falsity. We then obtain a result similar to Proposition 1, 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.

Example 3 For the first (deterministic) instance I in Example 2, the encoding $\overline{sat}(I)$ yields the least model

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

hence, $\overline{C}_M = \{b, c, d, h\}$, which gives rise to the super-control K^{+}_{M} such that $K^{+}_{M}(s) = \{\mathbf{a}\}$ for $s \in \{b, c, d\}$ and $K^{+}_{M}(s)$ is undefined for $s \in \{f, g, h\}$. In this case, there is a single control K refining K^{+}_{M}, which is given by $K(s) = \mathbf{a}$ for $s \in \{b, c, d\}$ discussed above. □

As computing a model of a Horn theory is a well-known polynomial problem (Dowling & Gallier 1984), we thus obtain the following result.

Theorem 2 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. (Dowling & Gallier 1984). 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 1 (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.

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 7 (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 $a \in \mathcal{A}_{ag} \cap poss(s)$ 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 (i) there is an a-path from s to E of length at most i, and (ii) from each state s' reachable from s , there is an a-path from s' to E of length at most k. 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) $s_i \Rightarrow \bigvee_{a \in \mathcal{A}_{ag} \cap poss(s)} s\_a_i$;

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

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

(3.3) for every $a \in \mathcal{A}_{ag} \cap poss(s)$, 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, but while doing it ignore the aspect (ii) of the meaning of s_i. The clauses in (3.1) 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 (expressed by 3.2). 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 3 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{ag} \subseteq \mathcal{A}$, 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 $a \in \mathcal{A}_{ag} \cap poss(s)$, 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 $K^+_M : \mathcal{S} \to 2^{\mathcal{A}_{ag}}$ 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. □

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.

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 any state $s \in \mathcal{S}$ and 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) $\left(\bigwedge_{a \in \mathcal{A}_{ag} \cap poss(s)} \overline{s\_a_i}\right) \Rightarrow \overline{s_i}$;

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

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

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

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

We thus obtain from Proposition3 easily the following result, which is the main result of this section so far.

Theorem 4 Let I consist of a system $A = (\mathcal{S}, \mathcal{A}, \Phi, poss)$, a set $\mathcal{A}_{ag} \subseteq \mathcal{A}$, 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 $\overline{\ell}_M(s) = \min\{j \mid M \not\models \overline{s\_a_j}, a \in \mathcal{A}_{ag} \cap poss(a)\}$. 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

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

k-maintains S w.r.t. E. □

Corollary 5 Problem k-MAINTAIN is solvable in polynomial time. □

Example 4 Continuing Example 2, for the nondeterministic variant I_1 where $\Phi(c, \mathbf{a}) = \{d, f\}$ instead of $\{d\}$, the formula $\overline{sat}'(I_1)$ is found unsatisfiable for any $k \geq 0$. On the other hand, for the instance I_2 where in addition $\Phi(g, \mathbf{a}') = \{f, h\}$ and $\mathbf{a}' \in poss(g)$, $\overline{sat}'(I_2)$ is satisfiable and has the least model

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

We thus obtain the super-control K^+_M defined on the states b, c, d, f, and g, where $K^+(s) = \{\mathbf{a}\}$ for $s \in \{c, d, f\}$ and $K^+(s) = \{\mathbf{a}'\}$ for $s \in \{b, g\}$. There is a single control K which refines K^+_M, namely $K(s) = \mathbf{a}$ for $s \in \{c, d, f\}$ and $K(s) = \mathbf{a}'$, for $s \in \{b, g\}$. □

Genuine procedural 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 $c[s]$ represents that $\overline{s_0}, \ldots, \overline{s_i}$ are assigned true; in particular, $i = -1$ represents that no $\overline{s_i}$ is assigned true yet. Similarly, value i for $c[s\_a]$ represents that $\overline{s\_a_1}, \ldots, \overline{s\_a_i}$ 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 $\mathcal{A}_{ag} \subseteq \mathcal{A}$
    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) For every $s$ in $E$, set $c[s] := -1$.
    (ii) For every $s$ in $\mathcal{S} \setminus E$, set $c[s] := k$ if $s \in S$ and
        $\mathcal{A}_{ag} \cap poss(s) = \emptyset$; otherwise, set $c[s] := 0$.
    (iii) For every $s$ in $\mathcal{S} \setminus E$ and $a \in \mathcal{A}_{ag} \cap poss(s)$, 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 states $s \in \mathcal{S}$ and $s' \in \Phi(a, s)$ where $a \in exo(s)$ and $c[s']{=}k$ do $upd(c[s], k)$.
    (ii) For any state $s \in \mathcal{S} \setminus E$,
        (a) if $s' \in \Phi(a, s)$ for $a \in \mathcal{A}_{ag} \cap poss(s)$ and $c[s']{=}i$
            such that $0 \leq i < k$ then do $upd(c[s\_a], i + 1)$.
        (b) if $\mathcal{A}_{ag} \cap poss(s) \neq \emptyset$ and $i{=} \min(c[s\_a] \mid a \in \mathcal{A}_{ag} \cap poss(s))$, then 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 $K : \mathcal{S} \setminus E \rightarrow \mathcal{A}_{ag}$ defined
    on all states $s \in \mathcal{S} \setminus E$ with $c[s] < k$ and such that
    $K(s) \in \{a \in \mathcal{A}_{ag} \cap poss(s) \mid c[s\_a] < k$ and $c[s\_a] = \min_{b \in \mathcal{A}_{ag} \cap poss(s)} c[s\_b]\}$.   □
```

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 6 Algorithm k-CONTROL solves problem k-MAINTAIN. Furthermore, for any input I it terminates in polynomial time. □

Encoding k-Maintainability for an Answer Set Solver

In this section, we show how computing a k-maintainable control can be encoded to a logic program, based on the results of the previous section. More precisely, we describe an encoding to non-monotonic logic programs under the Answer Set Semantics (Gelfond & Lifschitz 1991), which can be executed on one of the available Answer Set Solvers such as DLV (Eiter *et al.* 2000) or Smodels (Niemelä & Simons 1997). 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.

Input representation $F(I)$

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 $\mathcal{A}_{ag} {\subseteq} \mathcal{A}$ of agent actions is represented using a predicate `agent` by facts `agent(`a`)`, for each $a {\in} \mathcal{A}_{ag}$;
- 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 5 Coming back to Example 2, the input I is represented as follows:

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

□

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 $\overline{C}\_M$
 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 is a refinement of 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 6 The output of DLV for the input I and the above program, filtered to `control` is $\{$`control(b,a)`, `control(c,a)`, `control(d,a)`$\}$. This corresponds to the "maximal" control K mentioned earlier. □

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 $\overline{C}\_M$
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 alevel_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 7 The output of DLV for the input input I_2 and the above program, filtered to `control` is

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

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

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 compact 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 (finite) 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.

Computational Complexity

In this section, we give some results regarding the complexity of constructing k-maintainable controls under various assumptions.

We consider here the decision problem associated with k-MAINTAIN (deciding k-maintainability of S w.r.t. E in A, which we refer to as k-MAINTAINABILITY), and deciding the maintainability of S w.r.t. E in A, which we refer to as MAINTAINABILITY.

We first consider the problems in the setting where the constituents of an instance are explicitly given, i.e., the sets in enumerative form and the functions by their graphs in tables.

Theorem 7 Problem k-MAINTAINABILITY is **PTIME**-complete (under logspace reductions). Furthermore, the **PTIME**-hardness holds for 1-MAINTAINABILITY (i.e., k fixed to 1), even if all actions are deterministic and there is only one (deterministic) exogenous action.

Proof. (Sketch) The membership of k-MAINTAINABILITY in **PTIME** has been established above. The **PTIME**-hardness is shown by a reduction from deciding entailment of an atom q from a propositional Horn logic program π.

Briefly, the idea is to represent backward rule application through agent actions; i.e., for a rule r of $b_0 \leftarrow b_1, \ldots, b_m$, 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, $i \in \{1, \ldots, m\}$. In order to deal with cycles through rules, the states also carry level information. 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 is provable from π. Given that k rules exist in π, this is equivalent to k-maintainability of S w.r.t. E, and a k-maintaining control corresponds to a proof of q from π.

By using a special form of the rules in π and an exogenous action, it is possible with some coding tricks to emulate nondeterministic agent actions and sequences of agent actions by alternating sequences of 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. □

Theorem 8 MAINTAINABILITY is **PTIME**-complete. The **PTIME**-hardness holds even in absence of exogenous actions, or if all actions are deterministic and there is only one exogenous action.

Proof. (Sketch) The membership in **PTIME** is immediate from the previous theorem and the fact that S is maintainable w.r.t. E iff S is k-maintainable w.r.t. E for $k = |\mathcal{S}|$. The **PTIME**-hardness for the stated restrictions is shown by reductions similar to the one in the proof of Theorem 7. □

The problems are thus of the same difficulty as Horn SAT, which is **PTIME**-complete (Papadimitriou 1994). In some cases, the complexity is lower:

Theorem 9 Problem MAINTAINABILITY for systems with only deterministic actions and no exogenous actions is **NLOG-SPACE**-complete.

Proof. (Sketch) In this case, it can be shown that the problem amounts to deciding whether each $s \in S$ reaches some $s' \in E$ in the graph whose nodes are all states in $\mathcal{S}$ and which has an edge $s \rightarrow s'$ whenever $s' \in \Phi(s, a, s')$ for some $a \in \mathcal{A}_{ag} \cap poss(s)$. Since deciding reachability of a node s' from a node s in a directed graph is well-known **NLOG-SPACE**-complete (Papadimitriou 1994), the result follows. □

Another case in which the complexity is lower is if the maintenance phase is short and exogenous actions are excluded.

Theorem 10 Problem k-MAINTAINABILITY for systems without exogenous actions is in **LOG-SPACE**, if k is constant.

Proof. (Sketch) In this case, the problem consists in deciding whether for every state $s \in S$, some state in E is reachable within k steps by executing appropriate actions. More precisely, define inductively

$$\begin{aligned} r_0(s) &= s \in E, \\ r_{i+1}(s) &= s \in E \vee \exists a \in \mathcal{A}_{ag} \cap poss(s) \\ &\qquad \forall s' \in S(s' \in \Phi(s, a) \Rightarrow r_i(s')), \end{aligned}$$

for $i \geq 0$. Then, in absence of exogenous actions k-MAINTAINABILITY is equivalent to $r_k(s)$ for all $s \in S$. This can be checked, for constant k, in logarithmic workspace. □

Thus, in the two cases above, the problems can be reduced to deciding reachability of a node t from a node s in graph, where in the latter each node has at most one outgoing edge.

State descriptions by variables

We note that under compact state representation as described above, in which a system state s is represented by a vector $s = (v_1, \ldots, v_m)$ of values for fluents $f_1, \ldots, f_m$ ranging over given finite domains, the complexity of the problem increases by an exponential in general.

In this setting, we assume that the following membership predicates, evaluable in polynomial time, are available: $in\_Phi(s, a, s')$, $in\_poss(s, a)$, and $in\_exo(s, a)$ respectively for $s' \in \Phi(s, a, s')$, $a \in poss(s)$, and $a \in exo(s)$, respectively. Furthermore, $in\_S(s)$ and $in\_E(s)$ for deciding whether $s \in S$ and $s \in E$, respectively. We then obtain the following result.

Theorem 11 k-MAINTAINABILITY and MAINTAINABILITY are **EXPTIME**-complete, when the input is given in the compact representation from above. The **EXPTIME**-hardness holds even for 1-MAINTAINABILITY (i.e., k fixed to 1) and also for MAINTAINABILITY, even if all actions are deterministic and there is only one exogenous action.

Proof. (Sketch) The membership in **EXPTIME** follows easily from unpacking the compact state representation to an explicit (enumerative) one, which leads to an exponential increase in the worst case, and which can be constructed in exponential time. On the enumerative representation, the problem can then be solved in polynomial time as shown above. In total, this means that the problem is solvable in exponential time.

The **EXPTIME**-hardness can be shown by a reduction from deciding inference of an atom from a Horn logic program with variables (a datalog program). The construction lifts a similar one for propositional programs, showing **PTIME**-hardness for the respective problems under enumerative representation, to the Datalog case. □

In absence of exogenous actions, under compact representation k-MAINTAINABILITY for constant k and MAINTAINABILITY are in **PSPACE** provided that for the latter problem all actions are deterministic.

A more detailed discussion of complexity issues, with full proofs of all results will be given in the extended paper.

Conclusion

In this paper, we presented a polynomial time algorithm to compute the control for k-maintainability, if one exists. We then analyzed the complexity of constructing such controls 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.

In other related work, (Jensen, Veloso & Bryant 2003) considers the somewhat opposite problem of developing policies that achieve a given goal assuming at most k interferences from the environment. A formal connection, if any, between this problem and k-maintainability remains open. Also, in recent years there have been several work, such as (Cimatti et al. 2003), on planning with non-deterministic actions, but in none of those papers, agents actions and exogenous actions are viewed separately.

Acknowledgments

We thank the reviewers for their helpful suggestions to improve this paper.

References

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

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation* 14–23.

Ceri, S., and Widom, J. 1990. Deriving production rules for constraint maintainance. In McLeod, D., et al., eds., *Proc. 16th Int'l Conference on Very Large Data Bases (VLDB'90)*, 566–577. Morgan Kaufmann.

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

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

Dowling, W., and Gallier, H. 1984. Linear time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1:267–284.

Drummond, M. 1989. Situation control rules. In *Proc. First Int'l Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, 103–113.

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

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

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

Gelfond, M., and Lifschitz, V. 1992. Representing actions in extended logic programs. In Apt, K., ed., *Joint Int'l Conf. and Symp. on Logic Programming*, 559–573. MIT Press.

Ginsberg, M. 1989. Universal planning: An (almost) universally bad idea. *AI magazine* 40–44.

Jensen, R.; Veloso, M.; and Bryant, R. 2003. Synthesis of fault tolerant plans for non-deterministic domains. *ICAPS'03 Workshop on planning under uncertainty and incomplete information*, 64-73.

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

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

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

Nakamura, M.; Baral, C.; and Bjareland, M. 2000. Maintainability: a weaker stabilizability like notion for high level control. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, 62–67.

Niemelä, I., and Simons, P. 1997. Smodels – an implementation of the stable model and well-founded semantics for normal logic programs. In Dix, J., et al., eds., *Proc. 4th Int'l Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'97)*, LNAI 1265, 420–429. Springer.

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

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