Reactive Maintenance Policies over Equalized States in Dynamic Environments*

Zeynep G. Saribatur[1], Chitta Baral[2], and Thomas Eiter[1]

[1] Technische Universität Wien, Austria
{zeynep,eiter}@kr.tuwien.ac.at
[2] Arizona State University, USA
chitta@asu.edu

Abstract. We address the problem of representing and verifying the behavior of an agent following a policy in dynamic environments. Our focus is on policies that yield sequences of actions, according to the present knowledge in the state, with the aim of reaching some main goal. We distinguish certain cases where the dynamic nature of the environment may require the agent to stop and revise its next actions. We employ the notion of maintenance to check whether a given policy can maintain the conditions of the main goal, given a respite from environment actions. Furthermore, we apply state clustering to mitigate the large state spaces caused by having irrelevant information in the states, and under some conditions this clustering might change the worst-case complexity. By preserving the behavior of the policy, it helps in checking for maintenance with a guarantee that the result also holds in the original system.

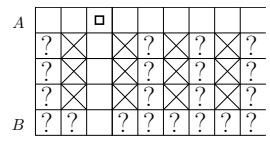
1 Introduction

Dynamic environments may change the state of the world and interfere with the behavior of a reactive agent that follows a given policy. During the execution of a plan given by the policy, a state change may require the agent to stop and examine the current situation, to determine the next steps. In such cases, rather than "achieving" certain conditions of a main goal, the focus is more on "maintaining" the conditions. Baral et al. [3] introduced maintenance given a *window of opportunity*, a respite from the environment actions. This notion enables us to distinguish the agent following a policy and doing its best to maintain the goal, if the environment does not interfere during a time period.

For an example, consider an agent that is looking for a person in a supermarket with a layout shown in Fig. 1(a). Although the agent knows the layout, it does not know where the person might be, and is given a policy (Fig. 1(b)) to follow. If the agent observes the person at any time step, then it stops and moves towards him. If the environment is static, i.e., the person does not move, then the agent's behavior following the policy can be represented as in [13]. However, our focus is on the dynamic nature of the environment; the person may also be moving, while the agent executes its actions. Thus, the environment actions play a role in the agent's behavior and need to be distinguished.

* This work has been supported by Austrian Science Fund (FWF) project W1255-N23, and Zeynep G. Saribatur's visit to ASU was supported by the Austrian Marshall Plan Foundation.

Fig. 1: Supermarket example



(a) agent's observation in a state

If at row A: walk towards right to the next aisle.
If reached the end of row A: walk towards row B.
If at row B: walk towards left to the next aisle.
If reached the end of row B: walk towards row A.
If observed the person: move towards the person.

(b) a policy

Different from [3], we are interested in policies that yield sequences of actions, which requires awareness of environment actions that may concurrently be made. Furthermore, while [3] considers explicit states, our focus is on implicit state representations, which allows for the use of logical formalisms to represent transitions. Moreover, the policies that we consider are defined using the representation power of logic programs, action theories or QBFs. Online planning and decision making are governed by these policies.

One concern of representing an agent's behavior for a given policy is the issue of keeping irrelevant information in the state which the policy does not use; having to represent such information adds to the state explosion problem. State clustering considered by [13] is a form of abstraction that omits such information, while preserving the behavior of the policy in static environments. We employ this notion by extending it to distinguish environment actions, which helps in defining the maintenance of a policy. Although such a clustering might not change the worst-case complexity in general, it makes a difference in practice, as one deals with information only related to the behavior. However, in certain situations the clustering may significantly decrease the complexity.

Our contributions are briefly summarized as follows:
(1) We extend the notion of maintainability to consider concurrent actions and focus on policies that yield sequences of actions (Sect. 3). We consider the possible outcomes of executing the latter in a dynamic environment. In our representation, we distinguish cases such as (a) the agent executes the actions with no interference, (b) the environment acts in a way that prevents the agent from executing the remainder of its actions, and (c) the agent realizes a way to reach the main goal, and stops executing its remaining actions. Case (c) can also occur in a static environment, but was not considered in [13].
(2) We introduce a state clustering that can distinguish environment movements and define a system that represents the policy's behavior in the dynamic environment (Sect. 4). We show that such a system preserves the relevant information, and helps in checking the maintenance with a guarantee that the result also holds in the original system (Sect. 5).
(3) We discuss complexity issues regarding the representation, such as checking for maintainability of a system for a given policy, and the effect of the clustering (Sect. 6). We also discuss the synthesis problem, i.e., constructing some maintenance policy.

2 Preliminaries

We define a system that represents dynamic environments as follows.

Definition 2.1 (System). *A* (dynamic) system *is a quadruple* $A{=}\langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, \Phi_c \rangle$*, where*
- $\mathcal{S}$ *is the finite set of states;*

- $\mathcal{S}_0 \subseteq \mathcal{S}$ *is the set of initial states;*
- $\mathcal{A} = \mathcal{A}_a \cup \mathcal{A}_e$ *is the finite set of agent* ($\mathcal{A}_a$) *and environment* ($\mathcal{A}_e$) *actions;*
- $\Phi_c : \mathcal{S} \times \mathcal{A}_a \times \mathcal{A}_e \to 2^{\mathcal{S}}$ *is a non-deterministic transition function.*

The idle action a_{nop} (resp. e_{nop}) is included in $\mathcal{A}_a$ (resp. $\mathcal{A}_e$) and Φ_c considers concurrent actions, where for all $s \in \mathcal{S}$, $\Phi_c(s, a_{nop}, e_{nop}) = \{s\}$. A sequence $\overline{a} = a_1, a_2, \ldots, a_n$ of agent actions is executable if

$$\exists s_0, \ldots, s_n : \forall i < n, s_{i+1} \in \Phi_c(s_i, a_{i+1}, e_{nop}) \ \wedge \ a_{i+1} \neq a_{nop}.$$

We denote such *(potential) plans* by Σ_a, and by $\Sigma_a(s)$ those that are executable from s. We use the notation $\Sigma'_a = \Sigma_a \cup \{a_{nop}\}$ to also consider the idle agent action.

We consider policies that have a main goal μ in mind, and guide the agent with action sequences that are computed according to the knowledge base KB, which is the formal representation of the world's model with a transition system view (as in [13]).

Definition 2.2 (Policy). *Given a system* $A = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, \Phi_c \rangle$ *and a set* Σ_a *of plans with actions of* $\mathcal{A}_a \subseteq \mathcal{A}$*, a* policy *is a function* $P_{\mu,KB} : \mathcal{S} \to 2^{\Sigma'_a}$ *s.t.* $P_{\mu,KB}(s) \subseteq \Sigma_a(s) \cup \{a_{nop}\}$.

For any state s, $\{a_{nop}\} \subseteq P_{\mu,KB}(s)$ should hold, for the cases of a moving environment while the agent is idle. We say that $P_{\mu,KB}$ is undefined for a state s if $P_{\mu,KB}(s) = \{a_{nop}\}$. For readability, we omit subscripts of P, as they are considered to be fixed.

Notice that a plan given by the policy might become inexecutable if the environment acts. In order to consider environments that may interfere with the agent's plan execution, we will express possible outcomes of the desire towards executing the *policy plans*.

3 Behavior of a policy in dynamic environments

We describe a system that represents possible outcomes of executing a policy plan in a dynamic environment, and define the maintenance by the policy.

Transitions as action sequences To consider execution of action sequences, we first extend the system $A = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{A}, \Phi_c \rangle$ to $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$, where $\Sigma = \Sigma'_a \cup \Sigma_e$ with $\Sigma_e = \mathcal{A}_e^*$ as the set of sequences of environment actions. The transition function $\Phi_\Sigma : \mathcal{S} \times \Sigma'_a \times \Sigma_e \to 2^{\mathcal{S}}$ yields the states resulting from executing concurrent action sequences: For $\overline{a} = \langle a_1, \ldots, a_n \rangle$ and $\overline{e} = \langle e_1, \ldots, e_n \rangle$,
$\Phi_\Sigma(s, \overline{a}, \overline{e}) = \{s' \mid \exists s_0, \ldots, s_n : \forall i < n, s_{i+1} \in \Phi_c(s_i, a_{i+1}, e_i) \ \wedge \ s_0 = s \wedge s_n = s'\}$;
it is undefined if $|\overline{a}| \neq |\overline{e}|$. We use $\Phi_\Sigma(s, P(s), \overline{e})$ as a shorthand for $\bigcup_{\overline{a} \in P(s)} \Phi_\Sigma(s, \overline{a}, \overline{e})$.

The evolution of the world described by the system is characterized by trajectories and the closure of a system is defined using these trajectories as follows.

Definition 3.1 (Trajectory and Closure). *In a system* $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$*, an alternating sequence of states and action sequences* $s_0, \sigma_1, s_1, \ldots, \sigma_n, s_n$ *is a* trajectory *if* $s_i \in \Phi_\Sigma(s_{i-1}, \overline{a}_i, \overline{e}_i), i \geq 0$*, for* $\sigma_i = (\overline{a}_i, \overline{e}_i) \in \Sigma$*. The* closure *w.r.t. a set* $S \subseteq \mathcal{S}$ *is* $Cl_\Sigma(S, A_\Sigma) = \bigcup_{s \in S} \{s_n \mid \exists$ *trajectory* $s_0, \sigma_1, s_1, \ldots, \sigma_n, s_n$ *in* $A_\Sigma, n \geq 0 : s_0 = s\}$.

Following the policy in a dynamic environment. We consider three outcomes of executing a policy plan in the dynamic environment:
(1) The environment's actions may not interfere with the execution of the plan, and the agent can execute the whole plan and reach the state that the policy was aiming for.

(2) The environment may act in a way that a state is reached, from which the remainder of the plan becomes non executable (even if the environment does no longer move).
(3) The agent may reach a state that has a possibility to reach the main goal, so that, instead of executing the remaining plan, a new plan can be determined towards the goal.

Formally, a transition function $\Psi_{P,\Sigma_e} : \mathcal{S} \times \Sigma'_a \rightarrow 2^{\mathcal{S}}$ yields the states by executing some plan returned by P:

$$\Psi_{P,\Sigma_e}(s,\overline{a}) \;=\; \{s' \mid \overline{a} \in P(s), \exists \overline{e} \in \Sigma_e : s' \in \Phi_\Sigma(s,\overline{a},\overline{e})\} \;\cup \tag{1}$$

$$\bigcup_{\overline{a}=\overline{a}'\overline{a}''\in P(s)} \{s' \mid \exists \overline{e} \in \Sigma_e, s' \in \Phi_\Sigma(s,\overline{a}',\overline{e}) \wedge\; \Phi_\Sigma(s',\overline{a}'',\overline{e}_{nop}) \models \bot\} \cup \tag{2}$$

$$\bigcup_{\overline{a}=\overline{a}'\overline{a}''\in P(s)} \{s' \mid \exists \overline{e} \in \Sigma_e, s' \in \Phi_\Sigma(s,\overline{a}',\overline{e}) \wedge\; \exists s'' \in \Phi_\Sigma(s',P(s'),\overline{e}_{nop}) : s'' \models \mu\} \tag{3}$$

where for a set S of states, $S \models \alpha \Leftrightarrow \forall s \in S,\, s \models \alpha$ and $\overline{e}_{nop} \in \{e_{nop}\}^*$. In (2) and (3), we focus on prefixes $\overline{a}'$ of $\overline{a}$ to compute the (middle) states reached while executing $\overline{a}$.

The states reachable from s if all of the plan $\overline{a}$ can be executed are computed in (1). The states in (2) are those reached due to some environment actions $\overline{e}$ during the execution of $\overline{a}$, where the remaining plan $\overline{a}''$ is no longer executable, even if the environment is idle after this point. From the middle states in (3), the main goal μ can be reached with a new policy plan (if the environment remains idle).

We represent the case when the environment remains idle with $\Psi_{P,\overline{e}_{nop}}$, where

$$\Psi_{P,\overline{e}_{nop}}(s,\overline{a}) = \{s' \mid \overline{a} \in P(s), s' \in \Phi_\Sigma(s,\overline{a},\overline{e}_{nop})\} \cup \bigcup_{\overline{a}=\overline{a}'\overline{a}''\in P(s)} \{s' \in \Phi_\Sigma(s,\overline{a}',\overline{e}_{nop}) \mid \exists s'' \in \Phi_\Sigma(s',P(s'),\overline{e}_{nop}) : s'' \models \mu\}.$$

Notice that $\Psi_{P,\overline{e}_{nop}}(s,\overline{a}) \subseteq \Psi_{P,\Sigma_e}(s,\overline{a})$.

From a state s, a state s' reached after trying to execute a plan $\overline{a}$, i.e., $s' \in \Psi_{P,\Sigma_e}(s,\overline{a})$, is referred as a *checkpoint state* from s, where the agent determines its next policy actions. The set of all such states from a set S of states is similar to Defn. 3.1 when the closure Cl_{P,Σ_e} is defined over the trajectories of Ψ_{P,Σ_e}.

Maintenance. The idea is to keep track of the checkpoint states when the policy is followed, and define the maintenance over them. First, the notion of unfolding a policy is defined as a sequence of states the system may go through if it follows the policy, while the environment remains idle, for at most some k steps.

Definition 3.2 (Unfold). *For a system $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$ and $s \in \mathcal{S}$, $Unfold_k(s, A_\Sigma, P)$ is the set of all sequences $\overline{s} = s_0, \ldots, s_l$ where $l \leq k$ and $s_0 = s$ s.t. $P(s_j)$ is defined for all $j < l$, $s_{j+1} \in \Psi_{P,\overline{e}_{nop}}(s_j, P(s_j) \backslash \{a_{nop}\})$, and if $l < k$, then $P(s_j)$ is undefined.*

Based on this, we define the k-maintainability.

Definition 3.3 (k-Maintainability). *For a system $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$, the policy P k-maintains $S \subseteq \mathcal{S}$ w.r.t. a goal condition μ, if for each state $s \in Cl_{P,\Sigma_e}(S, A_\Sigma)$ and sequence $s_0, s_1 \ldots, s_l$ in $Unfold_k(s, A_\Sigma, P)$ some $j \leq l$ exists s.t. $s_j \models \mu$.*

We say that the original system A_Σ is k-maintained by policy P w.r.t. μ, if P k-maintains the initial states S_0 w.r.t. μ.

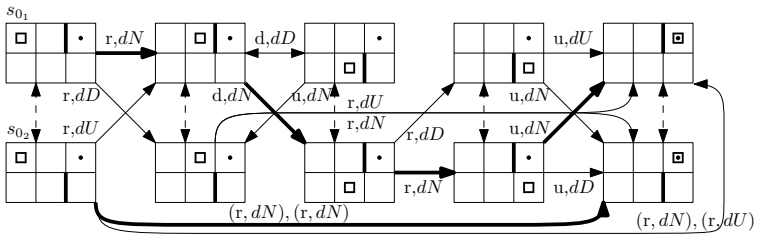


Fig. 2: Closure w.r.t. the initial states in the sliding door example

We illustrate over a simple example for better understanding of the concepts. We consider a basic scenario, since even a simplified (yet still interesting) supermarket example has quite a number of states and is difficult to visualize within the space limits.

Example 3.1. Consider a sliding door scenario, where an agent, initially located at (0,0), can move right (r), down (d) or up (u), and a sliding door, located between columns 1-2, can move up (dU), down (dD) or remain still (dN). The agent can look horizontally and detect if the door is located in its row. The main goal is to reach (0,2). Consider a policy P that tells to move right whenever possible and if not possible then to move up/down (depending on which one is executable). In case (0,2) is observable, the policy returns the plan to reach that cell. Once the agent reaches (0,2), no more action is taken.

Figure 2 shows possible trajectories from the initial states s_{0_1}, s_{0_2}, including those where the agent does not move (dashed arrows). All shown states constitute the closure w.r.t. the initial states according to the policy and the possible environment actions. The trajectories in which the environment remains still gives the unfolding trajectories from the initial states (thick arrows). One can see that the system is 4-maintained by the policy.

The door scenario is simplistic, and as one adds new properties of the agent, the environment, or a more involved policy, the state space immediately gets larger. Furthermore, as in the supermarket example, the state may contain information irrelevant to the agent's behavior, which leads to a large number of states with unnecessary information.

Note that if we restrict the concurrent action transition function Φ_c to only allow for execution of (a, e_{nop}) and (a_{nop}, e), this can model an alternating execution of agent and environment actions described in [3]. The transition (a_{nop}, e_{nop}) then corresponds to having no possible actions for the agent or the environment at a state. Thus, we have the following proposition when only policies with 1-step plans are considered.

Proposition 3.1 (Connection to Baral et al. [3]). *A system $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$, where Φ_Σ is built over a restricted Φ_c, is k-maintained by a 1-step policy P iff the corresponding system defined as in [3] is k-maintainable, due to existence of P.*

4 Omitting unnecessary information

State clustering by getting rid of irrelevant information w.r.t. the policy or the observability of the environment was considered in [13] with a focus on static environments, which guarantees any information gain in a state to hold in the successor state. This allows the state clusters to become more explicit as the agent traverses the environment.

However, in case of dynamic environments, such a clustering idea is unable to distinguish the environment's movement in a state's irrelevant/unobserved part. Fig. 3(a) shows some part of the system when the notion in [13] is applied to the supermarket example; the states where the agent observes the person are omitted for simplicity. The agent only knows that the environment did some actions $\overline{e}$. Therefore, whenever the person is not observed, the unobserved part is considered to be unknown, because the dynamic nature can not guarantee that some gained information holds in the next state.

To define maintenance, we must be able to distinguish the transitions where the environment does not move (especially, in the unobserved parts) and represent how this affects the knowledge about the state clusters. To this end, we use equalized d-states.

Definition 4.1. *An* equalized dynamic (d-) state *is a pair* $\langle \hat{s}, \hat{\theta} \rangle$*, where*
(1) the equalized state, $\hat{s}$*, contains the indistinguishable states w.r.t the policy, and*
(2) the inferred state, $\hat{\theta}$*, contains the states which are inferred to possibly hold by using the knowledge of the environment's movements.*

The state $\hat{s}$ contains the information relevant to the policy or the observability of the environment, while the state $\hat{\theta}$ makes further inferences to represent the effect of the environment's movements; in particular, whether the environment moved or not. Thus, there can be multiple pairs with identical equalized states, but different inferred states.

Building the clusters We consider two *classification functions* described by surjections:
- $h : S \rightarrow \Omega$, where Ω is the set of possible equalized states, and
- $h_r : S \rightarrow 2^{\Theta}$, where Θ is the set of possible inferred states.

Necessarily, h and h_r should satisfy that for every $\langle \hat{s}, \hat{\theta} \rangle$, we have $h_r^{-1}(\hat{\theta}) \subseteq h^{-1}(\hat{s})$. A state may be mapped to more than one inferred state cluster, as these clusters depend on the previous states and the movement of the environment.

The classification function h is based on the notion of indistinguishability, as in [13]. The state clustering is done only to omit the irrelevant information w.r.t the policy, so that P returns the same output for the cluster. Formally, the clustering satisfies the condition

$$\forall s \in \mathcal{S}, P(s) = P(h(s)) \tag{4}$$

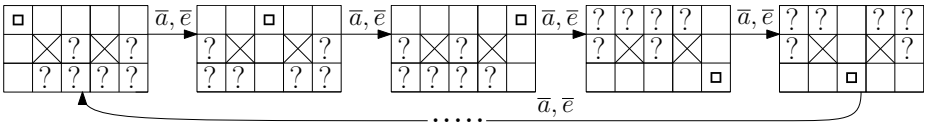
which makes sure that for states that are mapped to the same cluster, the policy returns the same plans, i.e., $\forall d, e \in \mathcal{S}, h(d) = h(e) \Rightarrow P(d) = P(e)$.

As said, inferred states depend on the previous states and the taken environment actions. In detail, the initial set of inferred states is $\Theta_0 = \{h(s) \mid s \in S_0\}$, and the clustering satisfies the constraint: for all $d, e \in \mathcal{S}$ such that $h_r(d) = h_r(e)$ it holds that $h(d) = h(e)$ and $\exists d', e'$: $d \in \Psi_{P,\Sigma_e \setminus \overline{e}_{nop}}(d', P(d')), e \in \Psi_{P,\Sigma_e \setminus \overline{e}_{nop}}(e', P(e'))$ with $h_r(d') = h_r(e')$. In other words, only the states that can be reached from a previous state, due to some sequence of environment actions, are mapped into the inferred states. This is similarly done for the case of $\overline{e}_{nop}$, to distinguish the states reached only by $\overline{e}_{nop}$.

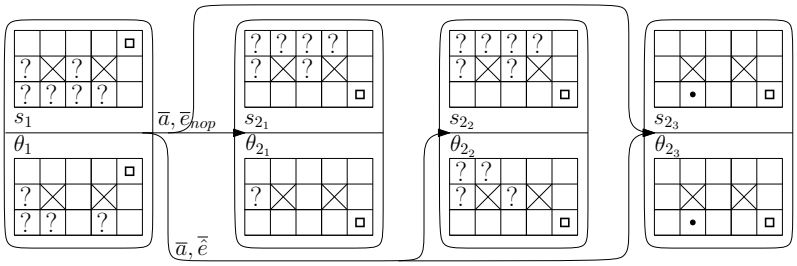
How to do state clustering h for action languages was shown in [13]. Inferred state clustering h_r is possible along the same lines, but not elaborated here due to space limits.

Abstract environment actions Clustering the states by omitting the information about the irrelevant part of the state leads to abstracting the irrelevant environment actions. Since the main aim is to represent whether or not the environment concurrently moved, we distinguish between the actions e_{nop} and $\hat{e}$ which is an abstraction of all other environment actions; we let $\widehat{\mathcal{A}}_e = \{e_{nop}, \hat{e}\}$ and consider the mapping $\phi_h : \mathcal{A}_e \rightarrow \widehat{\mathcal{A}}_e$.

Fig. 3: State clusters that distinguish environment actions



(a) Equalization is unable to distinguish the unobserved environment movements



(b) Pair states with inferred states that distinguish if the environment moved or not

Such an abstraction can be seen as the coarsest one possible, as it only distinguishes whether the environment moved or not. It is sufficient for defining maintenance, since the focus is on cases in which the environment does not move.

Example 4.1. Fig. 3(b) shows an example of equalized d-states. As expected, the inferred states have less possible locations for the person than the equalized states, since the possible locations are inferred more precisely depending on whether he/she moved or not. Furthermore, whether or not the person concurrently moves results in different inferred states. From state $\langle s_1, \theta_1 \rangle$, $\overline{e}_{nop}$ causes the cells observed at θ_1 to remain the same in θ_{2_1}, although to the agent's view, s_{2_1}, they become unknown. If the person executes some actions $\overline{\hat{e}}$, his possible locations in θ_{2_2} are inferred from θ_1. Notice that θ_{2_2} is not the same as s_{2_2}, as it shows the locations the person can move to in the same time steps as the agent. A transition may also result in the agent observing the person; then $s_{2_3}{=}\theta_{2_3}$ holds, since the person is obviously not in the unobserved parts.

5 Equalized dynamic systems

A system that represents the policy execution in a dynamic environment is defined over the original system by taking the classification functions and the policy into account.

Definition 5.1. *An* equalized dynamic system A_P^{h,h_r}*, w.r.t. the classification functions* h, h_r *and the policy* P*, is defined as* $A_P^{h,h_r} = \langle \widehat{\mathcal{S}}, \widehat{\mathcal{S}}_0, \Sigma'_a, \widehat{\Sigma}_e, \widehat{\Psi}_{P,\Sigma_e} \rangle$*, where*

- $\widehat{\mathcal{S}}$ *is the finite set of equalized d-states;*
- $\widehat{\mathcal{S}}_0 \subseteq \widehat{\mathcal{S}}$ *is the set of initial equalized d-states, if* $h^{-1}(\widehat{\theta}) \cap \mathcal{S}_0 \neq \emptyset$*;*
- Σ'_a *is the union of the set* Σ_a*, possible plans with agent actions* $\mathcal{A}_a$*, and* $\{a_{nop}\}$*;*
- $\widehat{\Sigma}_e{=}\widehat{A}_e^*$ *is the set of sequences of abstract environment actions (for* $\widehat{A}_e{=}\{e_{nop}, \hat{e}\}$*);*
- $\widehat{\Psi}_{P,\widehat{\Sigma}_e}{:}\,\widehat{\mathcal{S}}{\times}\Sigma'_a{\to}2^{\widehat{\mathcal{S}}}$ *is the policy transition function in the dynamic environment, i.e.,*

$$\widehat{\Psi}_{P,\widehat{\Sigma}_e}(\langle \hat{s}, \hat{\theta} \rangle, \overline{a}) = \{\langle h(s'), h_r(s') \rangle \mid \overline{a} \in P(\hat{s}), \exists s \in h_r^{-1}(\hat{\theta}), s' \in \Psi_{P,\Sigma_e}(s, \overline{a})\}.$$

The transitions $\widehat{\Psi}_{P,\overline{e}_{nop}}$ show an idle environment. For simplicity, $\langle \hat{s}, \hat{\theta} \rangle$ is denoted as $\hat{s}\hat{\theta}$.

The equalized dynamic system is defined over the state clusters of the checkpoint states in the original system. Reduced number of states and transitions help to focus on the details important for the policy, without losing any property of the behavior.

The closure $\widehat{Cl}(\widehat{S}, A_P^{h,h_r})$ is defined akin to Defn. 3.1, using the trajectories of $\widehat{\Psi}_{P,\widehat{\Sigma}_e}$.

Lemma 5.1. *For a given set S of states, any state s in $Cl_{P,\Sigma_e}(S, A_\Sigma)$, has a corresponding state $\hat{s}\hat{\theta}$ in $\widehat{Cl}(\widehat{S}, A_P^{h,h_r})$, where $\hat{s} = h(s)$ and $\hat{\theta} = h_r(s)$.*

The result holds since for every pair of successor states in A_Σ, there is a corresponding pair of equalized d-states in A_P^{h,h_r}. So any sequence of states considered in the closure of A_Σ w.r.t. a set S, has a corresponding sequence in the closure of A_P^{h,h_r} w.r.t. $\widehat{S}$.

Maintenance over the equalized dynamic system. We define the unfolding of the policy over the equalized d-states, similar to Defn 3.2, by considering $\widehat{\Psi}_{P,\overline{e}_{nop}}$. The clustering condition (4) ensures that no transitions will be introduced different from how the policy behaves in the original system. Furthermore, it ensures the following result.

Lemma 5.2. *For some k, for each sequence $s_0, \ldots, s_l$ in $Unfold_k(s, A_\Sigma, P)$, some sequence $\hat{s}\hat{\theta}_0, \ldots, \hat{s}\hat{\theta}_l$ in $\widehat{Unfold}_k(\hat{s}\hat{\theta}, A_P^{h,h_r})$ exists with $\hat{s}_i{=}h(s_i)$ and $\hat{\theta}_i{=}h_r(s_i), 0 \leq i \leq l$.*

We define the k-maintainability of the equalized dynamic system as follows, where $\hat{s}\hat{\theta} \models \mu \Leftrightarrow \forall s \in h^{-1}(\hat{s}) : s \models \mu$.

Definition 5.2 (Equalized k-Maintainability). $A_P^{h,h_r} = \langle \widehat{\mathcal{S}}, \widehat{\mathcal{S}}_0, \Sigma'_a, \widehat{\Sigma}_e, \widehat{\Psi}_{P,\Sigma_e} \rangle$ *is* k-maintainable*, if P k-maintains $\widehat{\mathcal{S}}_0$ w.r.t. μ: For each state $\hat{s}\hat{\theta}$ in $\widehat{Cl}(\widehat{S}_0, A_P^{h,h_r})$ and sequence $\hat{s}\hat{\theta}_0, \hat{s}\hat{\theta}_1, \ldots, \hat{s}\hat{\theta}_l$ in $\widehat{Unfold}_k(\hat{s}\hat{\theta}, A_P^{h,h_r})$ some $j \leq l$ exists s.t. $\hat{s}\hat{\theta}_l \models \mu$.*

Example 5.1. In the supermarket example for environment size 5×9 (Fig. 1(a)), the given policy 9-maintains the set of initial states. At any state in the closure, the person may be located at the farthest possible point, and if he does not move for at least 9 steps, then the agent will eventually observe the person and catch him.

The following theorem shows that the clustering does not introduce false positives.

Theorem 5.1 (Soundness). *If A_P^{h,h_r} is k-maintainable, then A_Σ is k-maintained by P.*

Proof. Assume that A_Σ is not k-maintained by P. Let $s \in Cl_{P,\Sigma_e}(S_0, A_\Sigma)$ be a state and $\tau{=}s_0, s_1 \ldots, s_l$ in $Unfold_k(s, A_\Sigma, P)$ such that $s_l \not\models \mu$. By Lemma 5.1-5.2, we know that $\exists \hat{s}\hat{\theta} \in \widehat{Cl}(\widehat{S}_0, A_P^{h,h_r})$ with $\hat{s} = h(s)$ and $\exists \hat{\tau}{=}\hat{s}\hat{\theta}_0, \hat{s}\hat{\theta}_1 \ldots, \hat{s}\hat{\theta}_l$ in $\widehat{Unfold}_k(\hat{s}\hat{\theta}, A_P^{h,h_r})$ with $\hat{s}_i{=}h(s_i), 0 \leq i \leq l$. By assumption on s_l, $\hat{s}\hat{\theta}_l \not\models \mu$. Hence, A_P^{h,h_r} is not k-maintainable.

In order to have completeness, we need further restrictions on the state clustering h, to avoid introducing spurious trajectories. We consider the properness condition [13]

$$\hat{s}\hat{\theta}' \in \widehat{\Psi}_{P,\widehat{\Sigma}_e}(\hat{s}\hat{\theta}, \overline{a}) \iff \forall s' \in h^{-1}(\hat{s}'), \exists s \in h^{-1}(\hat{s}) : \Psi_{P,\Sigma_e}(s, \overline{a}) \tag{5}$$

which ensures that if A_P^{h,h_r} has a transition from $\hat{s}\hat{\theta}_1$ to $\hat{s}\hat{\theta}_2$, then any state mapped to $\hat{s}\hat{\theta}_2$ has a transition from some state mapped to $\hat{s}\hat{\theta}_1$. This allows for the possibility of backtracking any trajectory found in A_P^{h,h_r} and map it back to A_Σ (see details in [13]).

Theorem 5.2 (Completeness). *If A_Σ is k-maintained by P and h is proper, then A_P^{h,h_r} is k-maintainable.*

The equalized dynamic system can represent static environments by only allowing e_{nop} and can be related to the equalized static system [13]. Assuming that the actions are reversible (as in the supermarket example), if the agent's observations during a plan execution contribute to the decision making in the next state, then we get the following.

Corollary 5.1. *If the equalized dynamic system A_P^{h,h_r} is k-maintainable, then the policy P works in at most k steps in the equalized static system $A_{h,P}$.*

The result follows, since the assumptions help in emulating (3) (for $\overline{e}_{nop}$ case) in the static setting. If the agent reaches a state while also observing the main goal on the way, then the policy will have the agent reach the main goal in the next state. However, the reverse of the corollary may not hold, as the dynamic nature of the environment may have the agent end up in a state that was not considered in the static environment.

6 Computational Complexity

In this section, we consider the computational complexity of k-maintainability.

Assumptions We assume that given states $s, s' \in \mathcal{S}$, which are given in a binary encoding, and $(a, e) \in \Sigma'_a \times \Sigma_e$, deciding $\Phi_c(s, (a, e), s')$ is in Σ_i^p, for some $i \geq 0$; this reflects theory-based specification by action theories, logic programs, or QBFs possibly with projective auxiliary variables. Checking executability of any sequences $\bar{a}$, $\bar{e}$ on $\mathcal{A}_a^*$, resp. $\mathcal{A}_e^*$, at a state s is thus in Σ_j^p, where $j = \max(1, i)$ (and complete for Σ_j^p, under mild assumptions). Consequently, deciding whether a sequence $\bar{a} \in \mathcal{A}_a^*$ may occur in $P(s)$ at all (thanks to a suitable $\bar{e}$) has the same complexity; we (reasonably) assume that $P(s)$ selects among those $\bar{a}$ only polynomially many and of polynomial length (in the state size), called *p-jump* plans, and that recognizing plans $\bar{a}$ in $P(s)$ and deciding $P(s) \neq \{a_{nop}\}$ is feasible in polynomial time (this holds e.g. if $P(s)$ is computable in logspace). Assuming that the goal test $s \models \mu$ is also in polynomial time, we obtain:

Lemma 6.1. *Deciding (i) given $s, \bar{a}$ and s' whether $\Psi_{P,\Sigma_e}(s, \bar{a}, s')$ holds is in Σ_{j+1}^p and (ii) given an action sequence $\bar{s}$ whether $\bar{s} \in \mathit{Unfold}_k(s, A_\Sigma, P)$ is in Σ_j^p.*

The increase to Σ_{j+1}^p in (i) is due to part (2) of $\Psi_{P,\Sigma_e}(s, \bar{a})$. The lemma also holds for goal tests in Π_i^p. If the initial state check $s \in \mathcal{S}_0$ is in Π_j^p, we obtain the following result.

Theorem 6.1 (k-Maintaining Check). *Deciding whether a system $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$ is k-maintained, $k \geq 0$, by a given policy P w.r.t. a goal μ is* PSpace*-complete.*

To see this, deciding $s \in Cl_{P,\Sigma_e}(\mathcal{S}_0, A_\Sigma)$ is by Lemma 6.1.(i) in NPSpace. We can thus check the existence of a counterexample to k-maintenance (i.e., a state s violating Defn 3.3) in NPSpace, where we guess the sequence $\bar{s} = s_0, s_1, \ldots, s_l$ in $\bar{s} \in \mathit{Unfold}_k(s, A_\Sigma, P)$ stepwise. As NPSpace = PSpace, this yields the upper bound. On the other hand, the problem is PSpace-hard already in plain settings, with deterministic actions and simple policies, due to the PSpace-completeness of succinct graph reachability [1]. The complexity is lowered, if we assume that for $s \in Cl_{P,\Sigma_e}(\mathcal{S}_0, A_\Sigma)$?

an oracle in some class of the Polynomial Hierarchy is available and that k is polynomially bounded; in particular, for a Σ_i^p oracle, we obtain then membership in Π_j^p.

In contrast, the complexity of *synthesizing* a k-maintaining policy P is much harder, even if we require that P returns only p-jump plans.

Theorem 6.2 (k-Maintaining Synthesis). *Deciding if a system $A_\Sigma = \langle \mathcal{S}, \mathcal{S}_0, \Sigma, \Phi_\Sigma \rangle$ is k-maintained, $k \geq 0$, by any (p-jump) policy P w.r.t. a goal μ is* NExpTime*-complete.*

The membership part follows as we can guess in exponential time a (polynomial-jump) policy P and check in polynomial space whether P is indeed k-maintaining. The NExpTime-hardness is shown by encoding a polynomial-space branching Turing Machine (BTM) [8] into 1-maintainability of a system A_Σ; a BTM is like an alternating TM, but in a special state it moves according to a *branching instruction* that depends on the whole configuration, not only on the current scanned symbol. The acceptance problem of such BTMs is NExpTime-complete. Informally, the policy (resp. environment) mimics the existential and branching (resp. universal moves).

Equalization. We assume that given s and $\hat{s}$ (resp. $\hat{\theta}$), deciding $s \in h^{-1}(\hat{s})$ ($s \in h_r^{-1}(\hat{\theta})$) is in Σ_ℓ^p, for some $\ell \geq 0$, where $|\Theta|$ is polynomial in the number $|\mathcal{S}|$ of states. In this setting, the transition function $\widehat{\Psi}_{P,\widehat{\Sigma}_e}(\hat{s}\hat{\theta}, \bar{a}, \hat{s}'\hat{\theta}')$ is decidable in $\Sigma_{j'}^p$, where $j' = \max(j, \ell) + 1$; thus, k-maintaining policy checking and policy synthesis have the same complexity as in the unequalized case in general, i.e., in Theorems 6.1 and 6.2.

The worst-case complexity drops if the equalization does an exponential compression, i.e., $|\Omega| = \mathcal{O}(\log |\mathcal{S}|)$ and $|\Theta| = \mathcal{O}(\log |\mathcal{S}|)$. Such a compression can, for instance, result by projecting away auxiliary fluents from the models of a logic program or SAT formula that serve to encode state constraints, if the number of admissible states is polynomially bounded. Each such state induces two clusters with all auxiliary fluent interpretations that admit resp. do not admit this state; all other interpretations form a further cluster.

Under exponential compression, A_P^{h,h_r} has a polynomial-size state set $\widehat{\mathcal{S}}$, and along the discussion above the complexity of policy checking drops to $\Pi_{j'}^p$; thus policy synthesis is in $\Sigma_{j'+1}^p$. However, the problems share a smaller upper bound that is tight.

Theorem 6.3 (Compression). *For $A_P^{h,h_r} = \langle \widehat{\mathcal{S}}, \widehat{\mathcal{S}}_0, \Sigma'_a, \widehat{\Sigma}_e, \widehat{\Psi}_{P,\Sigma_e} \rangle$ with exponential compression, both deciding if (i) A_P^{h,h_r} is k-maintainable w.r.t. μ and (ii) some (p-jump) policy P' exists s.t. $A_{P'}^{h,h_r}$ is k-maintainable w.r.t. μ is $\Theta_{j'}^p$-complete where $k \geq 0$.*

Here $\Theta_{j'}^p = \Delta_{j'}^p[\log n] = \mathsf{P}_{\|[c]}^{\Sigma_{j'-1}^p}$ are the problems decidable in polynomial time with logarithmically many $\Sigma_{j'-1}^p$ oracle calls, or equivalently, a fixed number c of rounds of *parallel* $\Sigma_{j'-1}^p$ oracle calls [7, 14]. Not that for $i = \ell = 0$, we obtain Θ_2^p, as $j = \max(1, i)$.

Informally, by one such round we can construct the 1-step transition graph with nodes $\widehat{\mathcal{S}}$ and edges $\hat{s}\hat{\theta} \to \hat{s}'\hat{\theta}'$ with label (a, e) if some $s \in h_r^{-1}(\hat{\theta})$ and $s' \in \Phi_c(s, (a, e))$ exist with $h(s') = \hat{s}'$ and $h_r(s') = \hat{\theta}$. The k-maintainability of policy P in (i) can then be checked, using a labeling technique on the graph and reachability tests, in polynomial time modulo goal tests $\hat{s}\hat{\theta} \models \mu$; the latter can be computed separately before with parallel $\Sigma_{j'-1}^p$ oracle queries, and likewise the plans in $P(s)$.

The existence of some P' in (ii) is decided similarly, where the graph and the goal test results are passed to an NP oracle that guesses and checks P'; this establishes $\Theta_{j'}^p$-

membership. Matching hardness is shown by a reduction from a suitable problem on QBFs: given $\mathrm{QBF}_{\exists,j'-1}$ instances $\Phi_1, \ldots, \Phi_m$, is the number of satisfiable ones even?

Remark No bound on k is imposed in this result; this is because polynomially many steps in the size of $\widehat{\mathcal{S}}$ will be sufficient to construct a counterexample for k-maintainability.

7 Discussion and Conclusion

In this paper, we have extended and combined the notions of equalization [13] and maintenance [3] to represent and verify the behavior of an agent following a policy in a dynamic environment. Equalization was extended to also consider inferred states according to projected movements of the environment, which helps in establishing maintenance. A more sophisticated behavior of the environment than executing any sequence of actions can easily be embedded in the current representation of the system. Maintenance in [3] was generalized to have concurrent actions and policies with sequences of actions that are executed under richer behavior patterns.

The policies described in [13] determine targets with some target function and call a planner for a conformant plan that guarantees reaching the target. In this paper, plans need not be conformant, as no targets are considered and all states reached by trying to execute the plan occur in the closure, and thus matter for defining the maintenance.

Our emphasis is mainly on the verification aspect, i.e., verifying if a given policy achieves desired properties. Such a policy testing and, in case it fails, refining with typically small changes would be preferred over a radical change. It also fits in the policy engineering life cycle. Eventually, we aim to bring in parameterization and verify policies in all possible sizes (up to some limit) of the environment, e.g., varying number of aisles in the supermarket. Our representation can easily be adjusted to such extensions.

Related Work Constructing agent control functions with various tasks such as "achievement" and "maintenance" is analyzed by [15], and [10] present a method to synthesize reactive plans based on linear temporal logic. In our work, we explicitly distinguish agent and environment actions to define maintenance, and the window of opportunity for the agent. We also focus on the verification aspect, and consider policies in the style of [13]. For further discussion on related work concerning maintenance, we refer to [3].

Plan verification [4] is on checking executability and goal achievement of a given plan. We focus on a more reactive case where, depending on the current situation, rather short sequences of actions are provided by the policy, and the aim is to eventually reach the goal while the agent traverses the environment with such guidance. Such an approach comes in handy especially if there is partial observability and a dynamic environment.

Abstraction can be useful to reason over the agent's behavior [2] or to do planning [9]. Over-approximation could achieve a significant reduction of the state space. However, trajectories found in the abstract system may be spurious, which would require further abstraction refinement. In this paper, we have focused on state clustering that is a faithful form of abstraction and gets rid of irrelevant information. This allows to focus on how the policy behaves in the system, and especially if the policy is not working properly, to detect this without having to consider all possible irrelevant states. Dealing with proper (non-faithful) abstractions remains as our future work.

The notion of irrelevant information and its effects were analyzed for planning in [12], in which different heuristics were introduced to omit such information. In our case, the irrelevancy is inferred from the given policy (e.g., target determination formulas in [13]) and which information it is making use of in determining the plans.

Verification of agent behavior represented using situation calculus action theory and of multi-agent systems has been studied e.g. in [6, 16] and [11], respectively. We focus on a single agent that follows the guidance of a given policy, and also consider the decision making of the policy or distinguish possible environment actions. Goal-driven agents acting in dynamic environments has been modeled with considering activities (i.e., sequences of actions meant to achieve a goal) [5], while the issue of having irrelevant information in the states was not the focus.

References

1. Balcázar, J.L.: The complexity of searching implicit graphs. AIJ 86(1), 171–188 (1996)
2. Banihashemi, B., De Giacomo, G., Lespérance, Y.: Abstraction in situation calculus action theories. In: Proc. of AAAI. pp. 1048–1055 (2017)
3. Baral, C., Eiter, T., Bjäreland, M., Nakamura, M.: Maintenance goals of agents in a dynamic environment: Formulation and policy construction. AIJ 172(12), 1429–1469 (2008)
4. Behnke, G., Höller, D., Biundo, S.: On the complexity of HTN plan verification and its implications for plan recognition. In: Proc. of ICAPS. pp. 25–33 (2015)
5. Blount, J., Gelfond, M., Balduccini, M.: A theory of intentions for intelligent agents. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) Proc. of LPNMR. pp. 134–142 (2015)
6. De Giacomo, G., Lespérance, Y., Patrizi, F., Vassos, S.: LTL verification of online executions with sensing in bounded situation calculus. In: Proc. of ECAI. pp. 369–374. IOS Press (2014)
7. Eiter, T., Gottlob, G.: The Complexity Class $\mathbf{\Theta_2^p}$: Recent Results and Applications in AI and Modal Logic. In: Proc. of FCT. pp. 1–18. No. 1279 in LNCS, Springer (1997)
8. Eiter, T., Lukasiewicz, T., Predoiu, L.: Generalized consistent query answering under existential rules. In: Baral, C., Delgrande, J., Wolter, F. (eds.) Proc. of KR. pp. 359–368 (2016)
9. Hoffmann, J., Sabharwal, A., Domshlak, C.: Friends or Foes? an AI planning perspective on abstraction and search. In: ICAPS. pp. 294–303 (2006)
10. Kabanza, F., Barbeau, M., St-Denis, R.: Planning control rules for reactive agents. AIJ 95(1), 67–113 (1997)
11. Lomuscio, A., Michliszyn, J.: Verification of multi-agent systems via predicate abstraction against ATLK specifications. In: Proc. of AAMAS. pp. 662–670 (2016)
12. Nebel, B., Dimopoulos, Y., Koehler, J.: Ignoring irrelevant facts and operators in plan generation. In: European Conference on Planning. pp. 338–350. Springer (1997)
13. Saribatur, Z.G., Eiter, T.: Reactive policies with planning for action languages. In: Michael, L., Kakas, A. (eds.) Proc. of JELIA, LNCS, vol. 10021, pp. 463–480. Springer (2016)
14. Wagner, K.: Bounded Query Classes. SIAM Journal on Computing 19(5), 833–846 (1990)
15. Wooldridge, M., Dunne, P.E.: Optimistic and disjunctive agent design problems. In: International Workshop on Agent Theories, Architectures, and Languages. pp. 1–14 (2000)
16. Zarrieß, B., Claßen, J.: Decidable verification of Golog programs over non-local effect actions. In: Proc. of AAAI. pp. 1109–1115 (2016)