Computing Exact Minimal Representations

I. The minimal representation problem

We are interested in obtaining agents that perform optimally on a task involving uncertainty while being as "simple" as possible, as measured by the size of their internal "representation". While this is a problem relevant in many fields, our motivation comes from the fields of robotics and computer vision. Embodied agents have sensors providing very high-dimensional data; the world's state is similarly high dimensional. Representing the agent's *belief* about the world is complicated enough, and often inference is intractable. However, it has been observed that, for certain tasks, it is not necessary to represent and plan using the entire belief.

Therefore, it is interesting to understand whether it is possible to solve a planning problem using a smaller representation than the entire belief. We will show that it is indeed possible and we will provide a constructive algorithm to obtain "representation-minimal" agents.

Let us make this idea more precise by introducing some notation. The following is the standard definition of POMDP.

Definition 1. A discrete-time POMDP is a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{Y}, T, \Omega, R, p_0, \gamma \rangle$ where $\mathcal{X}$ is the set of states; $\mathcal{U}$ is the set of commands; $\mathcal{Y}$ is the set of observations; $T : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ describes the conditional transition probabilities, such that $T(\boldsymbol{x}', \boldsymbol{x}, \boldsymbol{u}) = p(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u})$; $\Omega : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ describes the conditional observation probabilities, such that $\Omega(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})$; $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, so that $R(\boldsymbol{x}, \boldsymbol{u})$ is the reward obtained when choosing the command $\boldsymbol{u}$ when in state $\boldsymbol{x}$. The value $\gamma \in (0, 1]$ is the discount factor. Finally, $p_0 \in \mathsf{Measures}(\mathcal{X})$ is the initial prior on the state.

The problem is to find an agent that chooses the commands $\boldsymbol{u}_k$ as a function of the previous observations as to maximizes the discounted reward

$$\mathbb{E}\left\{\sum_{k=1}^{\infty} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{u}_k) \mid \boldsymbol{x}_0 \sim p_0\right\}. \tag{1}$$

Let us give a very general definition of what an "agent" is that highlights the role of the representation.

Definition 2 (Agent)**.** A (deterministic) agent is a tuple $\langle \Gamma, \boldsymbol{\gamma}_0, f, g \rangle$ where Γ is a set, $f : \Gamma \times \mathcal{Y} \to \Gamma$ is the agent dynamics, $g : \Gamma \times \mathcal{Y} \to \mathcal{U}$ is the agent policy, and $\boldsymbol{\gamma}_0 \in \Gamma$ is the initial state. The dynamics of the agent is given by

$$\begin{aligned} \boldsymbol{\gamma}_{k+1} &= f(\boldsymbol{\gamma}_k, \boldsymbol{y}_k), & (2)\\ \boldsymbol{u}_k &= g(\boldsymbol{\gamma}_k, \boldsymbol{y}_k). & (3) \end{aligned}$$

Thus, an agent is any causal mechanism that given a stream of observations $\boldsymbol{y}_k$ produces a stream of commands $\boldsymbol{u}_k$. The agent's "memory" is $\boldsymbol{\gamma}_k$ and Γ is called the "representation" or "state space". We will assume that all agents are deterministic for simplicity of exposition; however this restriction can be lifted.

Definition 3. An agent is *reward-optimal* if it maximizes the sum (1).

One particular optimal agent is the one whose representation Γ is the set of probability distributions $\mathsf{Measures}(\mathcal{X})$ over the state space $\mathcal{X}$ and $\boldsymbol{\gamma}_0 = p_0$. At each time the state of $\boldsymbol{\gamma}_k$ for this agent is the belief $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$. The transition $\boldsymbol{\gamma}_{k+1} = f(\boldsymbol{\gamma}_k, \boldsymbol{y}_k)$ equivalent of (2) is given by the Bayes filter, which recursively evolves the belief using the knowledge of the motion model $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k)$ and the observation model $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$. Finally, the mapping $\boldsymbol{u}_k = g(\boldsymbol{\gamma}_k, \boldsymbol{y}_k)$ is given by the policy $\pi^\star : \mathsf{Measures}(\mathcal{X}) \to \mathcal{U}$.

However, this is not the only optimal agent—in particular, there are agents that have a simpler "representation": rather than having as a state space the set $\mathsf{Measures}(\mathcal{X})$ they use a smaller set Γ. It is easy to see why propagating the belief over the entire state $\boldsymbol{x}$ can be wasteful: just imagine the case where most of the components of the state do not influence the reward. Thinking about the autonomous car example, the sensory data provided by a pair of cameras can easily reach 1 GB/s of "information"; however, most of this information is not "actionable information", for the simple reason that the decisions that must be taken at each moment (angle of the steering wheel, braking intensity, etc.) have much smaller dimensionality.

In this paper we want to compute the agent that has the smallest internal representation while still being optimal. (Section IV recalls other related notions of "simplicity".)

Definition 4. An agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g \rangle$ is *representation-minimal* if it is reward-optimal and has the smallest representation, measured with the cardinality $|\Gamma|$ of its state space Γ.

In this paper we consider the problem of finding representation-optimal agents automatically.

A. Coverage example

Let us introduce the example we use through the paper. We model the scenario of a robotic agent that must find a stationary intruder in a known environment; this is called a "coverage" problem. Let $G \subset \mathbb{Z}^2$ be a grid representing the environment. The state of the system is given by the robot position $q_k \in G$ and the pose of the intruder $e \in G$, so that $\mathcal{X} = G \times G$ and $\boldsymbol{x}_k = \langle q_k, e \rangle$.

We give the robot an ideal range-finder that returns the shape that that an observation $\boldsymbol{y}_k$ is a map $\boldsymbol{y}_k : \mathbb{Z}^2 \to \{□, ■, ▨\}$, where "□" means that the cell is free, "■" means that the cell is occupied, and "▨" means that the cell is behind one occluded cell. This is equivalent to observing the robot position q_k exactly, and to observe the intruder position if it is in the field of view.

The commands space $\mathcal{U}$ includes commands for moving and commands for declaring where the intruder is. The motion commands are $\mathcal{U}_{\mathrm{m}} = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$. The declaration commands are $\mathcal{U}_d = \{d_e\}_{e \in G}$. If the agent

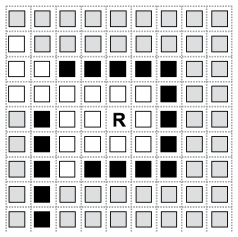
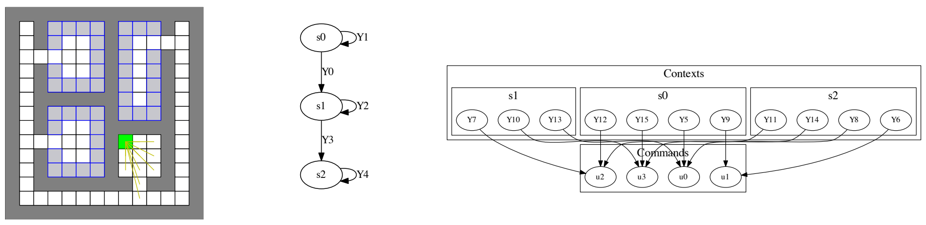


Figure 2. Sensing model



(a) Initial belief (b) Transitions f (c) Policy function g

Figure 1. The intruder problem. The example we use through the paper is one where a robot must explore a known environment to find a stationary intruder. In (a), the green square is the position of the robot. The blue area is the prior for the position of the intruder. The yellow rays show the robot's range-finder's field of view. For this particular example, we show that there exists an optimal agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g \rangle$ whose state space has dimension $|\Gamma| = 3$. (b) shows a representation of the agent's dynamics f. This is a state machine where s_i are the agent's states and the transitions are labeled by $Y_i \subset \mathcal{Y}$, which are subsets of observations. (c) shows a representation of the agent's policy g, which is a function from states $\times$ observations to the command space.

moves ($\boldsymbol{u} \in \mathcal{U}_m$), it gets a -1 reward. If the agent chooses
to declare ($\boldsymbol{u} \in \mathcal{U}_d$), then it gets a 0 reward if it has chosen the declaration of the correct cell, or $-\infty$ if it has chosen the wrong cell. Therefore, the optimal agent will explore until it is sure where the intruder is — either by direct observation or by exclusion.

We do not need to include observations noise or motion noise to make this example interesting. The only uncertainty comes from the fact that the intruder pose is unknown.

In this example, the belief space is a probability distribution over $G \times G$. Depending on the shape of the environment, the representation can be quite small. For example, in the scenario shown in Fig. 1 there exists an optimal agent that has only 3 states. There are also optimal agents that use 0 bits—the optimal policy is completely "reactive", in the sense that it can be expressed as a function of the current observations (Fig. 3).

II. A method for finding exact minimal representations

We describe an algorithm that, given a POMDP, computes an agent $\langle \Gamma, \gamma_0, f, g \rangle$ that is 1) reward-optimal (Def. 3); and 2) representation-minimal, as measured by the representation size $|\Gamma|$ (Def. 4). For example, in the scenario of Fig. 1, we show that there is an optimal optimal agent whose representation is minimal and uses $|\Gamma| = 3$ states.

A. Assumptions

The method involves first obtaining an optimal agent and then examining its input-output behavior to find an equivalent yet minimal agent. We make a series of assumptions:

- A We have a method to compute the optimal policy for the POMDP.
- B Under the optimal policy, there is a finite number of beliefs that are reached by the optimal agent.
- C The observations space $\mathcal{Y}$ is finite.
- D Either the optimal policy we have is deterministic, or the command space $\mathcal{U}$ is finite.

These last three, together, allow to compactly represent the input-output behavior of the agent through the histories of observations/commands.

We also make two temporary assumptions for ease of exposition:

- E Under the optimal policy, all trajectories have finite length.
- F The optimal policy is deterministic.

These two will be lifted later.

B. Overview

We first describe a method that works under assumptions E and F.

The method consists of two steps:

1) Step 1: Solving for the optimal policy $\pi^\star : \mathsf{Measures}(\mathcal{X}) \to \mathcal{U}$.
   In the first step we compute a solution to the POMDP. Any exact technique can be used. Because we regard this as a standard step, we relegate the details to the supplementary materials. In brief, following a standard construction, from the POMDP we create an MDP in belief space, and then we use policy iteration to obtain the optimal policy.
2) Step 2: Obtaining a minimal representation for carrying out the optimal policy. This step is the main contribution of this paper.
   a) Given the optimal policy, we enumerate all trajectories of the observations under the optimal policy and the corresponding optimal command. We obtain a map

$$\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}.$$

We call $\mathcal{T}$ a "decision table" because it is an exhaustive description of the input-output behavior of the optimal agent.

b) We run the createMinRep algorithm (Algorithm 1) with input $\mathcal{T}$. This is an iterative algorithm that at each iteration i adds one bit to the representation. The output after step i is a map

$$\mathcal{T}^{(i)} : \mathsf{Sequences}(\mathcal{Y} \times \{0,1\}^i) \to \mathcal{U} \times \{\mathsf{set}, \mathsf{nop}\}^i$$

where $\{0,1\}^i$ is the agent's candidate state space and $\{\mathsf{set}, \mathsf{nop}\}^i$ are extended commands that are related to the agent's internal candidate state dynamics.
After the algorithm terminates, the table $\mathcal{T}^{(i)}$ is a tabular representation of an agent with state space $\Gamma = \{0,1\}^i$ that is equivalent to the optimal agent.

C. Enumerating the system trajectories

At this point, we assume that we have an optimal policy $\pi^\star : \mathsf{Measures}(\mathcal{X}) \to \mathcal{U}$. Given this policy, we will construct an agent that has the minimal representation needed for carrying out the policy optimally. This means that this new agent will not have the representation $\Gamma = \mathsf{Measures}(\mathcal{X})$ but, rather, a much smaller set.

Given the policy $\pi^\star$, we can sample the POMDP trajectories that the agent sees when executing the policy. This is also a standard step (see Section A for details).

This gives us a set of sequences observations-commands of the type

$$\langle \langle \boldsymbol{y}_1, \boldsymbol{u}_1 \rangle, \langle \boldsymbol{y}_2, \boldsymbol{u}_2 \rangle, \dots, \langle \boldsymbol{y}_k, \boldsymbol{u}_k \rangle \rangle$$

which we call the "histories" $\mathcal{H}$.

Let us introduce some compact notation to talk about sequences. We use the standard notation "$\boldsymbol{y}_{1:k}$" to describe the tuple $\langle \boldsymbol{y}_1, \dots, \boldsymbol{y}_k \rangle$.

Given a set $\mathcal{Y}$, the set $\mathsf{Sequences}(\mathcal{Y})$ contains all subsequences of elements in $\mathcal{Y}$ of any length, including the empty sequence $\emptyset$. For example, if $\mathcal{Y} = \{\bigcirc, \triangle\}$, then $\mathsf{Sequences}(\mathcal{Y}) = \{\emptyset, \langle \circ \rangle, \langle \triangle \rangle, \langle \circ, \triangle \rangle, \langle \triangle, \circ \rangle, \dots\}$.

Therefore, the histories $\mathcal{H}$ are a subset of $\mathsf{Sequences}(\mathcal{Y} \times \mathcal{U})$.

From now on we can forget about the optimal policy or the original POMDP and we can work directly with the histories $\mathcal{H} \subset \mathsf{Sequences}(\mathcal{Y} \times \mathcal{U})$.

D. Tabulating the decisions table

Given the histories $\mathcal{H} \subset \mathsf{Sequences}(\mathcal{Y} \times \mathcal{U})$, we are interested in the agent's decisions. Therefore, we want to write the commands chosen by the agent at time k as a function of all the past observations up until time k.

For example, for $k = 3$, for a trajectory $\langle \langle \boldsymbol{y}_1, \boldsymbol{u}_1 \rangle, \langle \boldsymbol{y}_2, \boldsymbol{u}_2 \rangle, \langle \boldsymbol{y}_3, \boldsymbol{u}_3 \rangle \rangle$, we would record all these "decisions":

$$\begin{array}{rcl} \langle \boldsymbol{y}_1 \rangle & \mapsto & \boldsymbol{u}_1 \\ \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle & \mapsto & \boldsymbol{u}_2, \\ \langle \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \rangle & \mapsto & \boldsymbol{u}_3. \end{array}$$

Note that each row of this table is a map from the observations $\boldsymbol{y}_{1:k}$ to the last commands $\boldsymbol{u}_k$. However, the table contains also all subsequences. If we have n trajectories of length K, then we will record Kn rows.

Thus from the histories $\mathcal{H} \subset \mathsf{Sequences}(\mathcal{Y} \times \mathcal{U})$ we have recovered the "decision table" $\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}$. This is an equivalent representation of the policy and thus of the optimal agent. From now on, we can forget about the histories $\mathcal{H}$ and work with the decision table $\mathcal{T}$.

Example 5. For example, suppose that $\mathcal{Y} = \{\circ, \triangle\}$, $\mathcal{U} = \{\blacksquare, \bigstar\}$ and the histories $\mathcal{H}$ consist of two trajectories shown below. Then the decision table $\mathcal{T}$ has 5 elements.

Histories $\mathcal{H} \subset \mathsf{Sequences}(\mathcal{Y} \times \mathcal{U})$	Decision table $\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}$
$\mathcal{H} = \begin{cases} \langle\langle \bigcirc, \blacksquare\rangle, \langle \triangle, \blacksquare\rangle, \langle \bigcirc, \bigstar\rangle\rangle, \\ \langle\langle \triangle, \bigstar\rangle, \langle \triangle, \bigstar\rangle\rangle. \end{cases}$	$\mathcal{T} : \begin{cases} \langle \bigcirc \rangle & \mapsto \blacksquare, \\ \langle \bigcirc, \triangle \rangle & \mapsto \blacksquare, \\ \langle \bigcirc, \triangle, \bigcirc \rangle & \mapsto \bigstar, \\ \langle \triangle \rangle & \mapsto \bigstar, \\ \langle \triangle, \triangle \rangle & \mapsto \bigstar. \end{cases}$

E. Characterization of optimal agents given $\mathcal{T}$

From the knowledge of the decisions $\mathcal{T}$, we can write down what conditions ensure that the choice of agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g\rangle$ is optimal.

Definition 6. An agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g\rangle$ *reproduces* the input-output behavior $\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}$ if: for all sequences $\boldsymbol{y}_{1:k}$ in the domain of $\mathcal{T}$, the decision $\boldsymbol{u}_k = \mathcal{T}(\boldsymbol{y}_{1:k})$ can be written as a function g of the last observations $\boldsymbol{y}_k$ and the representation, which is formally a function $\varphi : \mathsf{Sequences}(\mathcal{Y}) \to \Gamma$ of all the history up to $k-1$:

$$\mathcal{T}(\boldsymbol{y}_{1:k}) = g(\varphi(\boldsymbol{y}_{1:k-1}), \boldsymbol{y}_k).$$

In turn, we can write φ recursively using f and $\boldsymbol{\gamma}_0$, as $\varphi(\emptyset) = \gamma_0$ and

$$\varphi(\boldsymbol{y}_{1:k}) = f(\varphi(\boldsymbol{y}_{1:k-1}), \boldsymbol{y}_k).$$

Therefore, we can formalize the problem as follows:

Problem 7. *Given an input-output behavior $\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}$, find the agent $\langle \Gamma, \gamma_0, f, g\rangle$ that, among all possible agents that reproduce $\mathcal{T}$, has the minimal representation Γ.*

Note that at this point we have forgotten about the original state space; the original belief space; and the optimal policy; and we are working directly using $\mathcal{T}$.

F. Why do we need states at all?

The decision table $\mathcal{T} : \mathsf{Sequences}(\mathcal{Y}) \to \mathcal{U}$ describes the input-output behavior of an optimal agent. Given these decisions we will find a state-space description of an agent that can reproduce the same input-output behavior with an internal state space with the minimal cardinality.

Let's discuss the topic intuitively. In planning under uncertainty, we usually consider agents whose state space is the belief space. Therefore, we are used to think of the agent's state as follows:

The agent's state is a belief that summarizes the past observations.

While this is true (for those agents), do temporarely suppress this notion and let's think more from first principles.

Why do agents need a state at all?

Let us examine the extreme cases where a state is not needed.

The extreme case is if the optimal policy was to always execute the same action $\boldsymbol{u}^\star$. Then we could define an optimal agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g\rangle$ with trivial state space of only one state, by setting $\Gamma = \{\bot\}$, $\boldsymbol{\gamma}_0 = \bot$, $f(\boldsymbol{\gamma}_k, \boldsymbol{y}_k) = \bot$ and $g(\boldsymbol{\gamma}_k, \boldsymbol{y}_k) = \boldsymbol{u}^\star$.

The slightly less extreme case is where the optimal agent is "reactive", in the sense that the optimal command is a function $r : \mathcal{Y} \to \mathcal{U}$ of the current observation. Also in this case there is an optimal agent $\langle \Gamma, \gamma_0, f, g\rangle$ with a trivial state space with only one state: $\Gamma = \{\bot\}$, $\boldsymbol{\gamma}_0 = \bot$, $f(\boldsymbol{\gamma}_k, \boldsymbol{y}_k) = \bot$ and $g(\boldsymbol{\gamma}_k, \boldsymbol{y}_k) = r(\boldsymbol{y}_k)$.

Any policy more complicated than this, when the current observations do not determine univocally the commands, need the agent to have a state space, where the state keeps track of the relevant events happened in the past. In this interpretation, this is the meaning of state:

The function of the agent's state is to disambiguate the commands when they are not determined entirely by the current observations.

We will be constructing a state for an agent guided by this principle.

Algorithm 1 createMinRep

Input: A decision table $\mathcal{T}:\mathsf{Sequences}(\mathcal{Y})\to\mathcal{U}$,
Output: An agent $\langle\Gamma,\gamma_0,f,g\rangle$ that reproduces $\mathcal{T}$.

Data structure:
A decision table $\mathcal{T}^{(i)}:\mathsf{Sequences}(\underbrace{\mathcal{Y}\times\{0,1\}^i}_{\text{context } \mathcal{C}^{(i)}})\to\underbrace{\mathcal{U}\times\{\mathsf{set},\mathsf{nop}\}^i}_{\text{decision } \mathcal{D}^{(i)}}$

```
$\mathcal{T}^{(0)} \leftarrow \mathcal{T}$
for $i \leftarrow 0, 1, 2, \ldots$:
    # Compute the set of ambiguous contexts $\mathcal{C}_{\text{amb}} \subset \mathcal{D}^{(i)}$
    $\mathcal{C}_{\text{amb}} \leftarrow \{\boldsymbol{c} \in \mathcal{C}_{\text{all}} \mid |\mathcal{D}_{\boldsymbol{c}}| > 1\}$ .

    if $\mathcal{C}_{\text{amb}} == \emptyset$:
        return $\mathsf{createAgent}(\mathcal{T}^{(i)})$

    # Choose a context $\overline{\boldsymbol{c}} \in \mathcal{C}_{\text{amb}}$ to disambiguate
    $\overline{\boldsymbol{c}} \leftarrow \arg\min_{\boldsymbol{c}\in\mathcal{C}_{\text{amb}}} \mathsf{earliestdiv}(\boldsymbol{c})$

    # Choose two decisions that are ambiguous
    $\boldsymbol{d}_a, \boldsymbol{d}_b \leftarrow \min_{\boldsymbol{d}_a,\boldsymbol{d}_b\in\mathcal{D}_{\boldsymbol{c}}} \mathsf{trajdiv}_{\boldsymbol{c}}(\boldsymbol{d}_a, \boldsymbol{d}_b)$

    # Expand the representation to disambiguate $\overline{\boldsymbol{c}}$
    $\mathcal{T}^{(i+1)} \leftarrow \mathsf{expandRep}(\mathcal{T}^{(i)}, \overline{\boldsymbol{c}}, \boldsymbol{d}_a, \boldsymbol{d}_b)$
```

G. *Incremental computation of the minimal representation (*createMinRep*)*

We now sketch the algorithm createMinRep (Algorithm 1) that solves Problem 7: given a decision table $\mathcal{T}:\mathsf{Sequences}(\mathcal{Y})\to\mathcal{U}$ it creates an agent that reproduces $\mathcal{T}$ and has the minimal representation.

The agent returned by the algorithm has a state space composed by bits. The basic idea is to start from an empty representation with zero bits and incrementally grow Γ one bit at a time until it is complex enough to reproduce the optimal input-output behavior specified by $\mathcal{T}$.

createMinRep is an iterative algorithm. Each iteration i adds one bit to the state space, so that the candidate state space at iteration i is $\Gamma^{(i)}=\{0,1\}^i$. During the iterations, the columns of the decision table $\mathcal{T}$ are enlarged to include the new states trajectories alongside the observations trajectories. So, while the domain of the original $\mathcal{T}$ are the sequences of observations $\mathsf{Sequences}(\mathcal{Y})$, the domain of $\mathcal{T}^{(i)}$ are the sequences of observations and states $\mathsf{Sequences}(\mathcal{Y}\times\Gamma^{(i)})$.

Morever, at each iteration also the command space is enlarged with extra commands that act on the state. We call this command space $\{\mathsf{set},\mathsf{nop}\}^i$. The semantics of these commands is that one bit is set to "1" when the corresponding command is equal to "set", and it does not change otherwise. More formally, let the agent state be composed by i bits $\left\langle\gamma^1,\ldots,\gamma^i\right\rangle\in\{0,1\}^i$ and let $\left\langle v^1,\ldots,v^i\right\rangle\in\mathcal{V}^i$ be the corresponding commands. The state dynamics of the agent is given by

$$\gamma_{k+1}^i=\begin{cases}\gamma^i, & \text{if } v_k^i=\mathsf{nop},\\ 1, & \text{if } v_k^i=\mathsf{set}.\end{cases}$$

We call "context" the tuple $\langle\text{current state},\text{current observations}\rangle$. Define the "context space"

$$\mathcal{C}^{(i)}\triangleq\mathcal{Y}\times\{0,1\}^i.$$

Further, we call "decision" the tuple $\langle\text{physical command},\text{state-changing command}\rangle$. Define the "decision space"

$$\mathcal{D}^i\triangleq\mathcal{U}\times\{\mathsf{set},\mathsf{nop}\}^i.$$

With this notation, at iteration i the algorithm maintains a decision table

$$\mathcal{T}^{(i)} : \mathsf{Sequences}(\underbrace{\mathcal{Y} \times \{0,1\}^i}_{\text{context } \mathcal{C}^{(i)}}) \to \underbrace{\mathcal{U} \times \{\mathsf{set}, \mathsf{nop}\}^i}_{\text{decision } \mathcal{D}^{(i)}}.$$

When the algorithm converges, $\mathcal{T}^{(i)}$ contains an explicit description of the internal dynamics of the agents (the dynamics of the state $\{0,1\}^i$) as well as the map from state to commands.

At the first iteration of Algorithm 1 (line 1), $\mathcal{T}^{(0)}$ is set to the input table $\mathcal{T}$, the "context space" $\mathcal{C}^{(0)}$ is equal to the observation space $\mathcal{Y}$, and the "decision space" $\mathcal{D}^{(0)}$ is equal to the command space $\mathcal{U}$.

At each iteration (line 6) the algorithm checks whether the table $\mathcal{T}^{(i)}$ contains a complete nonambiguous description of an agent by computing the set of "ambiguous" contexts $\mathcal{C}_{\text{amb}}$. Let us introduce some nomenclature to define $\mathcal{C}_{\text{amb}}$.

Given a context $\boldsymbol{c}$ we define the set $\mathcal{D}_{\boldsymbol{c}}$ of decisions that must be taken for the context $\boldsymbol{c}$. Let $j \in \mathsf{traj}$ be an index ranging over the the set of all trajectories. Write the table $\mathcal{T}^{(i)}$ as a series of tuples composed by the context history $\boldsymbol{c}_{1:\circ}$, where "$\circ$" denotes the last index of the trajectory (rather than writing k^j), and the corresponding decision $\boldsymbol{d}_\circ$:

$$\mathcal{T}^{(i)} = \{\langle \boldsymbol{c}_{1:\circ}, \boldsymbol{d}_\circ \rangle^j \mid j \in \mathsf{traj}\},$$

Let $\mathcal{C}_{\text{all}}$ be the set of all contexts:

$$\mathcal{C}_{\text{all}} \triangleq \{\boldsymbol{c}_\circ^j \mid j \in \mathsf{traj}\} \subset \mathcal{C}^i. \tag{4}$$

For each context $\boldsymbol{c} \in \mathcal{C}_{\text{all}}$ define the set $\mathcal{D}_{\boldsymbol{c}}$ of all decisions taken when in $\boldsymbol{c}$:

$$\mathcal{D}_{\boldsymbol{c}} \triangleq \{\boldsymbol{d}_\circ^j \mid j \in \mathsf{traj} \wedge \boldsymbol{c}_\circ^j = \boldsymbol{c}\}. \tag{5}$$

Finally, define the set $\mathcal{C}_{\text{amb}} \subset \mathcal{C}^{(i)}$ of all contexts that are ambiguous:

$$\mathcal{C}_{\text{amb}} \triangleq \{\boldsymbol{c} \in \mathcal{C}_{\text{all}} \mid |\mathcal{D}_{\boldsymbol{c}}| > 1\}. \tag{6}$$

If $\mathcal{C}_{\text{amb}}$ is empty, it means that the decision table describes the state space evolution of an agent that reproduces the original input-output behavior. Therefore, we can create the agent from $\mathcal{T}^{(i)}$ (line 7). See Section II-J for a description of createAgent.

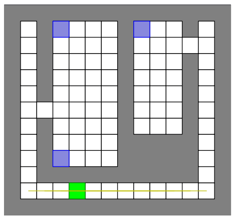
If $\mathcal{C}_{\text{amb}}$ is not empty, we need to increase the size of the representation. This is done by choosing one context $\overline{\boldsymbol{c}}$ in $\mathcal{C}_{\text{amb}}$ (line 10) as well as two ambiguous decisions $\boldsymbol{d}_a, \boldsymbol{d}_b \in \mathcal{D}_{\overline{\boldsymbol{c}}}$ to "disambiguate" (line 13). The context chosen is one that minimizes the function earliestdiv. This function makes us choose the context that can be disambiguated by introducing one bit as early as possible in the history of the agent. This is described later in Section II-K. (At a first read, these details can be skipped, though the details are important for establishing minimality.)

Given a context $\overline{\boldsymbol{c}}$ and the decisions $\boldsymbol{d}^a, \boldsymbol{d}^b \in \mathcal{D}_{\overline{\boldsymbol{c}}}$, the function expandRep extends the representation by adding one bit to the representation (line 16). Intuitively, the function adds a bit to the representation so that instead of having the context $\overline{c}$ mapping ambiguously to both $\boldsymbol{d}^a$ and $\boldsymbol{d}^b$, there will be two contexts $\langle \overline{\boldsymbol{c}}, 0 \rangle$ and $\langle \overline{\boldsymbol{c}}, 1 \rangle$ mapping nonambiguously to the $\boldsymbol{d}^a$ and $\boldsymbol{d}^b$. Moreover, it also extends the command space, by adding a command that is used to set the additional bit. The technical details for are described in Section II-I.

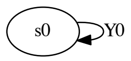
H. Examples

Before describing the algorithm's details, let us describe some example results.

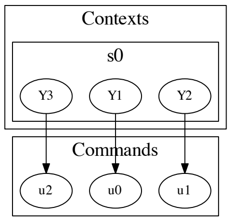
Some example results are shown in Fig. 3 and Fig. 4 for scenarios of different complexity. Fig. 3a shows a case in which the optimal agent is purely reactive. The agent will go into the room on the left. At that point it can see if the intruder is in one of those two locations. If it's not, then it is certain that it is in the other



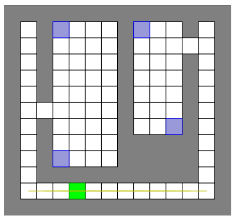
(a) Initial belief



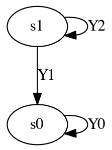
(b) Transitions f



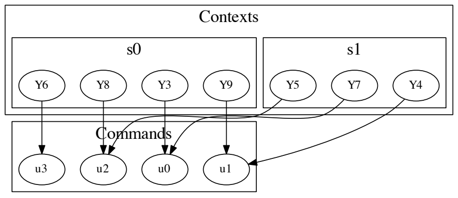
(c) Policy function g



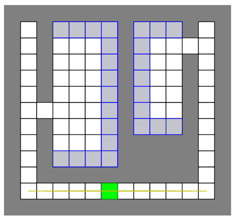
(d) Initial belief



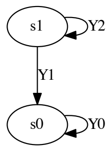
(e) Transitions f



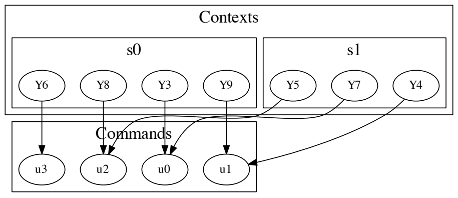
(f) Policy function g



(g) Initial belief



(h) Transitions f



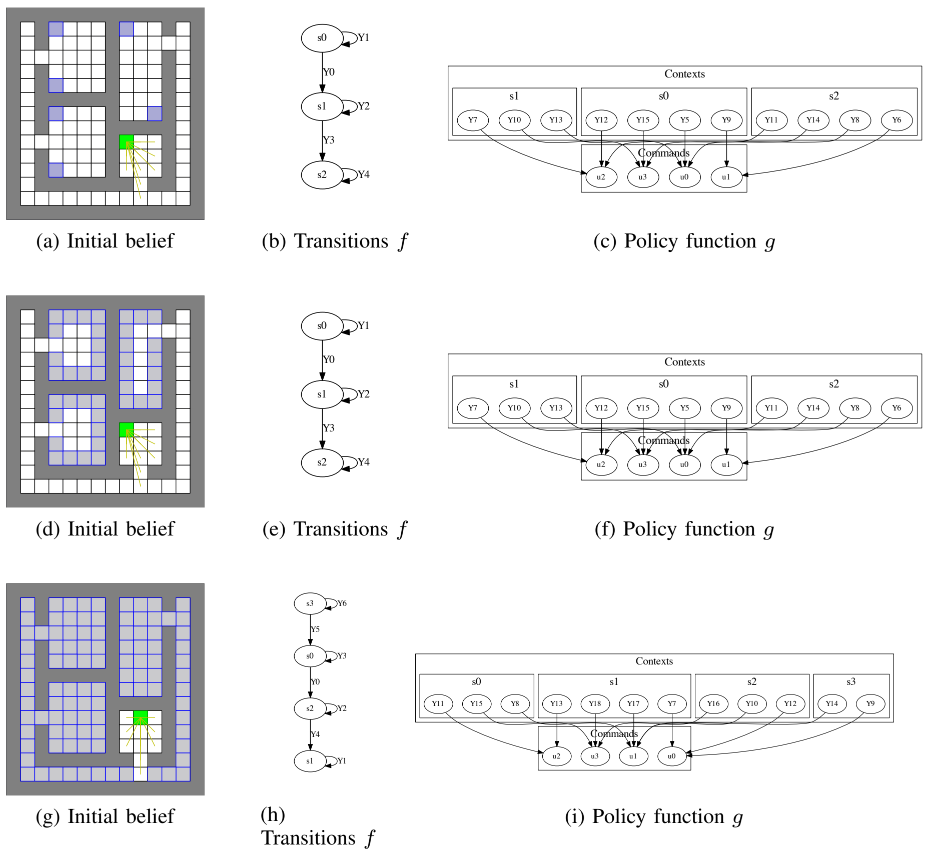
(i) Policy function g

Figure 3. Visualization of minimal agent. The first column shows the environment and the initial belief about the pose of the intruder. The second and third column show a visualization of the minimal agent found by our method. The second column shows the state space as well as the transition function f. The nodes are the states of the agent. The transitions are labeled by the symbols Y_i. These represents *subsets* of the possible observations $\mathcal{Y}$. The agent shown in the first row is purely reactive, having one state. The agent shown in the second and third row has a state space with two states. The third column shows a visualization of the function $g : \Gamma \times \mathcal{Y} \to \mathcal{U}$. This function takes a state and an observation into a command. We call the tuple formed by a state and an observations a "context". For example, in (c), the agent has only one state. Depending on which of the 3 obervations subsets are seen it uses one of three commands. For the policy visualized in (f), there are two states. Note that for each state, the partition of the observations space is different.

room. Therefore, the plan is completely open loop: go to the first room, and this plan can be executed without any memory. The agent's state space (Fig. 3b) has only one trivial state. Correspondingly, the policy is a function from observations to commands (Fig. 3c). For compactness, in these diagrams we represent subsets of observations $Y_i \subset \mathcal{Y}$.

Fig. 3d shows the case where the intruder has 4 possible locations in two rooms. In this case the optimal plan is to go first in the left room, and if the intruder is not there, go to the other room to disambiguate between the two remaining hypotheses. This plan needs one bit of memory. The bit can be interpreted as "I know no intruder was in the first room". The agent's has a state space with one transition (Fig. 3e). The transition happens when the agent sees that there is no intruder in the first room. The policy for the agent now depends on the agent's current state in addition to the current observations (Fig. 3f).

Note that the dimension of the representation depends on the complexity of the task, not the dimension of the belief space. For example, the following agent



(a) Initial belief (b) Transitions f (c) Policy function g

(d) Initial belief (e) Transitions f (f) Policy function g

(g) Initial belief (h) Transitions f (i) Policy function g

Figure 4. Visualization of minimal agents using 3 or 4 states. See the caption of Fig. 3 for an explanation of the diagrams.

has a more complicate belief space to keep track of (Fig. 3g), but has the same complexity of representation (Fig. 3h).

I. *Expanding the representation* (expandRep)

Algorithm 2 expandRep

Input:

- a decision table $\mathcal{T}$: Sequences$(\mathcal{C}) \to \mathcal{D}$
- a context $c \in \mathcal{C}_{\text{amb}}$
- two decisions $\boldsymbol{d}^a, \boldsymbol{d}^b \in \mathcal{D}_{\boldsymbol{c}}, \boldsymbol{d}^a \neq \boldsymbol{d}^b$

Output: a decision table $\tilde{\mathcal{T}}$: Sequences$(\mathcal{C} \times \{0,1\}) \to (\mathcal{D} \times \{\text{set}, \text{nop}\})$

The expandRep function is the core of the algorithm. It is given as input a decision table $\mathcal{T}$: Sequences$(\mathcal{C}) \to \mathcal{D}$ and an ambiguous context $\boldsymbol{c}$, together with two decisions $\boldsymbol{d}_a, \boldsymbol{d}_b \in \mathcal{D}_{\boldsymbol{c}}$. This means that in the decision table there are two trajectories ending in $\boldsymbol{c}$ which are associated to two decisions $\boldsymbol{d}_a \neq \boldsymbol{d}_b$. Let $\boldsymbol{c}^a_{1:\circ}$, $\boldsymbol{c}^b_{1:\circ}$ be two trajectories for which $\boldsymbol{c}^a_\circ = \boldsymbol{c}^b_\circ = \boldsymbol{c}$ and $\mathcal{T}(\boldsymbol{c}^a_{1:\circ}) = \boldsymbol{d}^a$ and $\mathcal{T}(\boldsymbol{c}^b_{1:\circ}) = \boldsymbol{d}^b$.

We are going to expand the representation so that the context $\boldsymbol{c}$ is expanded to $\boldsymbol{c}'_a = \langle \boldsymbol{c}, 0 \rangle$, mapping to $\boldsymbol{d}^a$, and $\boldsymbol{c}'_b = \langle \boldsymbol{c}, 1 \rangle$, mapping to $\boldsymbol{d}^b$.

Formally, we expand the context $\mathcal{C}$ to $\tilde{\mathcal{C}} = \mathcal{C} \times \{0,1\}$. Call γ be the additional element to the context $\tilde{\boldsymbol{c}}_k$.

$$\tilde{\boldsymbol{c}}^j_k = \langle \boldsymbol{c}^j_k, \gamma^j_k \rangle \in \mathcal{C} \times \{0,1\}.$$

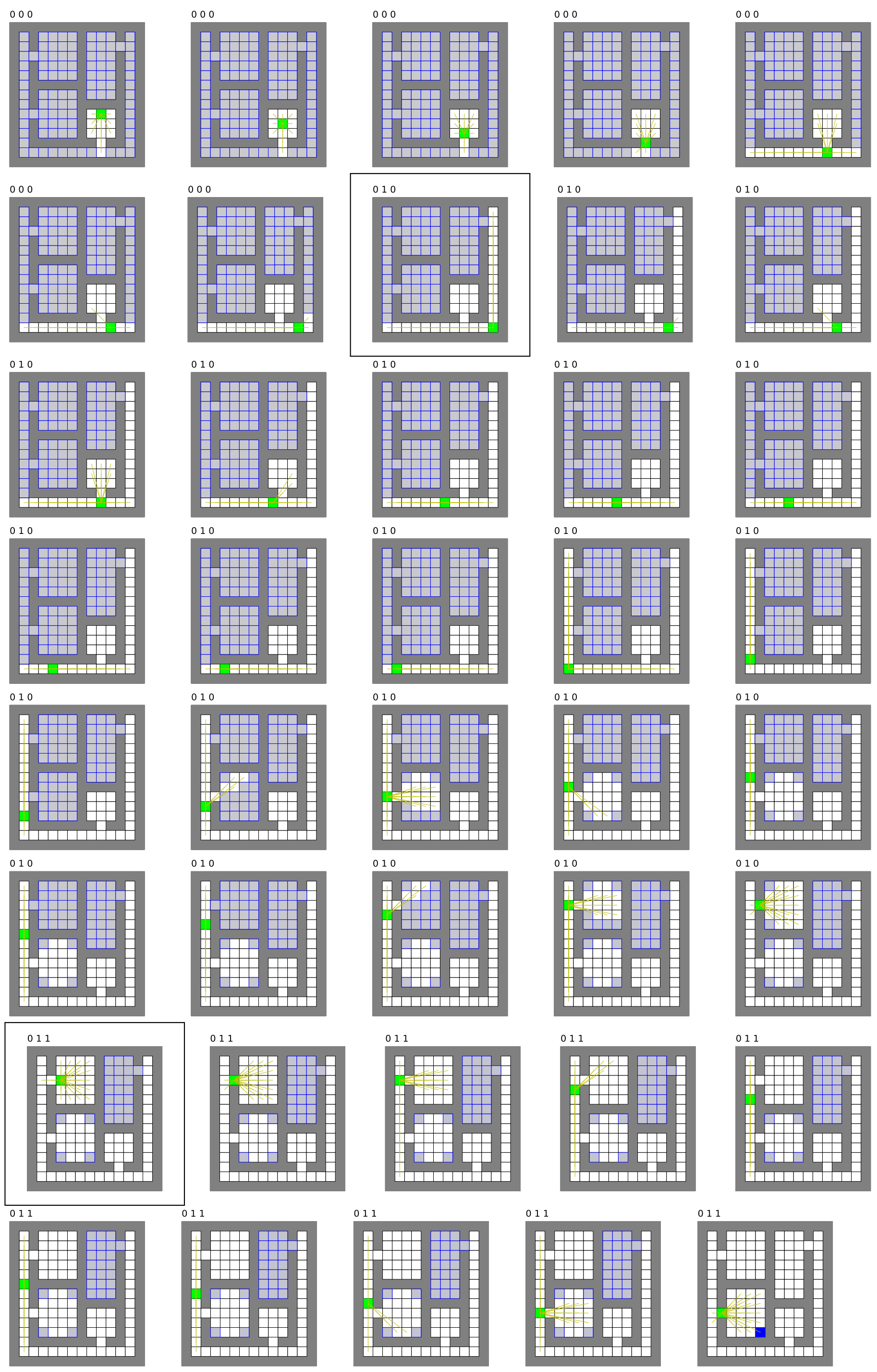


Figure 5. One trajectory of the reduced optimal agent, for the agent is visualized in Fig. 4h and Fig. 4i. On the top of each map the 3 bits that compose the state are shown. Note, however, that only 4 configurations are reachable so that the number of effective states is 4, not 8 (Fig. 4h). The two moments when the state changes are marked with a black border. In the first case, the state switches from $\langle 0, 0, 0\rangle$ to $\langle 0, 1, 0\rangle$. The agent has just discovered that there is no intruder in the rightmost corridor. In the second case, the state switches from $\langle 0, 1, 0\rangle$ to $\langle 0, 1, 1\rangle$. The agent has just cleared the room in the top left.

To obtain the new table $\tilde{\mathcal{T}}$ we will need to decide how to assign the value γ_k^j for all trajectories $j \in \mathrm{traj}$. We already want to set $\gamma = 0$ for $j = a$ and $\gamma = 1$ for $j = b$. Now, we want to decide on all other trajectories $j \in \mathrm{traj}$.

We will create a causal mechanism using an extended decision set. Formally, we expand the decisions $\mathcal{D}$ to $\tilde{\mathcal{D}} = \mathcal{D} \times \{\mathsf{set}, \mathsf{nop}\}$ and the decisions

$$\tilde{\boldsymbol{d}}_k^j = \langle \boldsymbol{d}_k^j, v_k^j \rangle \in \mathcal{D} \times \{\mathsf{set}, \mathsf{nop}\}.$$

The idea is to invoke the set command when the trajectories first differ. Let τ be the minimum time when the two trajectories $\boldsymbol{c}_{1:\circ}^a$ and $\boldsymbol{c}_{1:\circ}^b$ differ:

$$\tau = \min_k \{ \; k \;\mid\; \boldsymbol{c}_k^a \neq \boldsymbol{c}_k^b \}.$$

We thus set

$$v_k^j = \begin{cases} \mathsf{set}, & \text{if } j = b \text{ and } k = \tau. \\ \mathsf{nop}, & \text{otherwise.} \end{cases}$$

The idea is that the sequence $\boldsymbol{c}_{1:\tau}^b$ is the "triggering sequence" for this bit. Consequently we set γ to 1 for all trajectories and time that have seen that sequence:

$$\gamma_k^j = \begin{cases} 1, & \text{if } \boldsymbol{c}_{1:\tau}^b \text{ is a prefix of } \boldsymbol{c}_{1:k}^j, \\ 0, & \text{otherwise.} \end{cases}$$

J. *Creating the agent (*createAgent*)*

Algorithm 3 createAgent

---

input: A nonambiguous table $\mathcal{T}$: $\mathsf{Sequences}(\mathcal{Y} \times \{0,1\}^i) \to (\mathcal{U} \times \{\mathsf{set}, \mathsf{nop}\}^i)$.
output: An agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g \rangle$.

Here we give the details for creating the agent $\langle \Gamma, \boldsymbol{\gamma}_0, f, g \rangle$ from the decision table $\mathcal{T}$.

The state space for the agent is $\Gamma = \{0,1\}^i$, and $\boldsymbol{\gamma}_0$ is a vector of i zeros.

1) Constructing the agent's function g: From the entries of $\mathcal{T}$ we create the agent's function $g : \mathcal{Y} \times \Gamma \to \mathcal{U}$. The table maps sequences in $\mathcal{Y} \times \Gamma$ to decisions in $\mathcal{U} \times \{\mathsf{set}, \mathsf{nop}\}^i$. So the generic element can be written as

$$\langle \langle \boldsymbol{y}_{1:\circ}^j, \boldsymbol{\gamma}_{1:\circ}^j \rangle, \langle \boldsymbol{u}_\circ^j, v_\circ^j \rangle \rangle \in \mathsf{Sequences}(\mathcal{Y} \times \{0,1\}^i) \times \mathcal{U} \times \{\mathsf{set}, \mathsf{nop}\}^i.$$

We only care about the last time instant and the last command, thus obtaining

$$\langle \langle \boldsymbol{y}_\circ^j, \boldsymbol{\gamma}_\circ^j \rangle, \boldsymbol{u}_\circ^j \rangle \subset (\mathcal{Y} \times \{0,1\}^i) \times (\mathcal{U}).$$

By construction, this the graph of a function $g : \mathcal{Y} \times \{0,1\}^i \to \mathcal{U}$, which represents the agent's dynamics.

2) Constructing the agent's function f: Moreover, we argue that the subrelation

$$\langle \langle \boldsymbol{y}_\circ^j, \boldsymbol{\gamma}_\circ^j \rangle, v_\circ^j \rangle \subset (\mathcal{Y} \times \{0,1\}^i) \times (\{\mathsf{set}, \mathsf{nop}\}^i).$$

is a graph of a function $h : (\mathcal{Y} \times \{0,1\}^i) \to \{\mathsf{set}, \mathsf{nop}\}^i$. From this relation we construct the function $f : (\mathcal{Y} \times \{0,1\}^i) \to \{0,1\}^i$ which describes the dynamics $\boldsymbol{\gamma}_{k+1} = f(\boldsymbol{y}_k, \boldsymbol{\gamma}_{k+1})$. The dynamics can be written as

$$\gamma_{k+1}^i = \begin{cases} 1, & \text{if } h^i(\boldsymbol{y}_k, \boldsymbol{\gamma}_k^i) = \mathsf{set}, \\ \gamma_k^i, & \text{if } h^i(\boldsymbol{y}_k, \boldsymbol{\gamma}_k^i) = \mathsf{nop}. \end{cases}$$

K. Choosing how to increase the representation – earliestdiv($\boldsymbol{c}$)

The only detail left is the explanation of the heuristics for choosing the context to disambiguate.

By assumption the decision table is ambiguous, so there is at least one $\boldsymbol{c}$ such that $|\mathcal{D}_{\boldsymbol{c}}| > 1$. We will choose the context that minimizes a certain heuristics.

Fix an ambiguous context $\boldsymbol{c}$, and let $\mathcal{D}_{\boldsymbol{c}}$ be the commands associated to $\boldsymbol{c}$. Because $\boldsymbol{c}$ is ambiguous, there are at least two such decisions. For any two such decisions $\boldsymbol{d}_a, \boldsymbol{d}_b \in \mathcal{C}_{\boldsymbol{c}}$, we will define $\mathsf{firstdiv}_{\boldsymbol{c}}(\boldsymbol{d}_a, \boldsymbol{d}_b)$ as the time the corresponding trajectories first differ. We know that the table $\mathcal{T}$ contains the entries

$$\begin{aligned} \langle t_a \mid \boldsymbol{c} \rangle &\mapsto \boldsymbol{d}_b, \\ \langle t_b \mid \boldsymbol{c} \rangle &\mapsto \boldsymbol{d}_b. \end{aligned}$$

Rewrite the trajectories highlighting the common prefix p (possibly empty) and the first context to which they differ as follows:

$$\begin{aligned} \langle p \mid \boldsymbol{c}_a \mid \tau_a \mid \boldsymbol{c} \rangle &\mapsto \boldsymbol{d}_a, \\ \langle p \mid \boldsymbol{c}_b \mid \tau_b \mid \boldsymbol{c} \rangle &\mapsto \boldsymbol{d}_b. \end{aligned}$$

We take $|p|$ to be the value of $\mathsf{firstdiv}_{\boldsymbol{c}}(\boldsymbol{d}_a, \boldsymbol{d}_b)$:

$$\mathsf{firstdiv}_{\boldsymbol{c}}(\boldsymbol{d}_a, \boldsymbol{d}_b) = |p|.$$

For an ambiguous context $\boldsymbol{c}$, call earliestdiv($\boldsymbol{c}$) the time there is a divergence

$$\mathsf{earliestdiv}(\boldsymbol{c}) = \min_{\boldsymbol{d}_a, \boldsymbol{d}_b \in D_{\boldsymbol{c}}} \{\mathsf{firstdiv}_{\boldsymbol{c}}(\boldsymbol{d}_a, \boldsymbol{d}_b) \mid \boldsymbol{d}_a \neq \boldsymbol{d}_b\}. \tag{7}$$

At each iteration, we choose the $\boldsymbol{c}$ with the minimum value of earliestdiv($\boldsymbol{c}$) to disambiguate. This is important in the proof of optimality.

L. Analysis

We can prove the following points:

1) The algorithm createMinRep always terminates.
2) The agent returned by createMinRep reproduces the given decision table $\mathcal{T}$.
3) The agent returned by createMinRep has the minimal representation.

Points (1) and (2) are still true if the context to disambiguate is chosen randomly in $\mathcal{C}_{\text{amb}}$; however, point (3) regarding optimality depends on this precise choice of context as described in Section II-K.

to write

III. Relaxing the assumptions

We now relax assumption E: *Under the optimal policy, all trajectories have finite length.*

to write

IV. Discussion

to write

Appendix

A. *Solving the POMDP*

to write

B. *Enumerating the trajectories*

to write