Observation Reduction for Strong Plans

Wei Huang[1], **Zhonghua Wen**[1,2], **Yunfei Jiang**[1], **Lihua Wu**[1]
1. Software Research Institute, School of Information Science and Technology,
Sun Yat-sen University, Guangzhou, China
2. College of Information Engineering, Xiangtan University, Xiangtan, China
Email: huangbodao@yahoo.com.cn

Abstract

Strong planning under full or partial observability has been addressed in the literature. But this research line is carried out under the hypothesis that the set of observation variables is fixed and compulsory. In most real world domains, however, observation variables are optional and many of them are useless in the execution of a plan; on the other side, information acquisition may require some kind of cost. So it is significant to find a minimal set of observation variables which are necessary for the execution of a plan, and to best of our knowledge, it is still an open problem. In this paper we present a first attempt to solve the problem, namely, we define an algorithm that finds an approximate minimal set of observation variables which are necessary for the execution of a strong plan under full observability (i.e. a state–action table); and transforms the plan into a strong plan under partial observability (i.e. a conditional plan branching on the observations built on these observation variables).

1 Introduction

In recent years, increasing interest has been devoted to planning in non–deterministic domains. Strong planning (i.e. finding automatically a plan which is guaranteed to achieve the goal regardless of nondeterminism) is an important problem in this research field. Under full observability or partial observability, this problem has been addressed in the literature: in the frame for full observability proposed in [Cimatti *et al.*, 1998b; 2003], a plan is a state-action table; in the frame for partial observability proposed in [Bertoli *et al.*, 2001; 2006], a plan is a conditional plan branching on the values of observation variables.

The cases of full observability, see e.g., [Cimatti *et al.*, 1998b; 2003], where the whole status of the domain is observed at every step, and the cases of partial observability, see e.g., [Boutilier, 2002; Eiter *et al.*, 2003; Herzig *et al.*, 2003; Bertoli *et al.*, 2001; 2006], where the plan executor cannot access the whole status of the domain, are both under the assumption that the set of observation variables is fixed and compulsory. In real world domains (e.g., many robotics, control, and scheduling domains), however, observation variables are optional and many of them are useless in the executions of the plans; on the other side, information acquisition may require some kind of cost (e.g. time, money, and power etc.). So it is significant to find a minimal set of observation variables that are necessary for the execution of a plan, and to best of our knowledge, it is still an open problem.

In this paper we present a first attempt to solve the problem. Under the hypothesis that any two distinct states of the domain can be distinguished by an observation variable at least, we define an algorithm that finds an approximate minimal set (written as $\mathcal{V}_{obs}$) of observation variables that are necessary for the execution of a strong plan under full observability (i.e. a state-action table), and transforms the plan into a strong plan under partial observability (i.e. a conditional plan branching on the observations built on $\mathcal{V}_{obs}$). This algorithm first exploits the belief state [Bonet and Geffner, 2000] (i.e. the set of possible current states) at every step to compute $\mathcal{V}_{obs}$; then, it transforms the given state–action table into a conditional plan branching on the observations built on $\mathcal{V}_{obs}$. The algorithm is based on the frameworks for strong planning under full observability proposed in [Cimatti *et al.*, 2003] and partial observability proposed in [Bertoli *et al.*, 2006].

The paper is organized as follows. We first recap the frameworks proposed in [Cimatti *et al.*, 2003] and [Bertoli *et al.*, 2006] (Section 2); then, we introduce the key notions of the algorithm and describe the algorithm in detail (Section 3); finally, Section 4 draws some conclusions and discusses future research directions.

2 Domains, Observations, Plans, and Problems

In this section, we briefly review some definitions of strong planning in non-deterministic domains under full observability and partial observability that are relevant to our work. Further details and examples of these definitions can be found in [Cimatti *et al.*, 2003; Bertoli *et al.*, 2006].

A *planning domain* is a model of a generic system with its own dynamics.

Definition 1 *A* planning domain *is a tuple* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, *where* $\mathcal{S}$ *is a finite set of states,* $\mathcal{A}$ *is a finite set of actions, and* $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ *is the transition function. The transition function associates to each state* $s \in \mathcal{S}$ *and to each action* $a \in \mathcal{A}$ *the set* $\mathcal{R}(s, a) \subseteq \mathcal{S}$ *of next states.*

The set of actions that are applicable in state s is $\text{ACT}(s) = \{a | \exists s' \in \mathcal{R}(s,a)\}$.

In practice, a domain may have many observation variables, whose value can be observed at run–time.

Definition 2 *Let $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ be a planning domain. Let $\mathcal{V}$ be a finite set of* observation variables. *An* observation function *over $\mathcal{S}$ and $\mathcal{V}$ is a function $\mathcal{X} : \mathcal{S} \times \mathcal{V} \rightarrow \{\top, \bot\}$.*

From Definition 2, for each state, there is a corresponding complete set of assignments to observation variables.

Plans can control the evolutions of the domain by triggering actions. A plan under full observability is a set of *state–action* pairs.

Definition 3 *A* plan *π_F for Σ is a set of pairs $\langle s, a \rangle$ (i.e. a state–action table), where: $s \in \mathcal{S}$, $a \in \text{ACT}(s)$, and there is at most one action a such that $\langle s, a \rangle \in \pi_F$ for any state s.*

The execution of a state–action table π_F in Σ can be represented as *execution structure*, i.e. a directed graph in which the nodes are all of the states of the domain that can be reached by executing actions in the state–action table, and the arcs represent possible state transitions caused by actions in π_F.

Definition 4 *Let π_F be a state–action table for Σ. The* execution structure *induced by π_F from the set of initial states $\mathcal{I} \subseteq \mathcal{S}$ is a tuple $K = \langle \mathcal{Q}, \mathcal{T} \rangle$, where $\mathcal{Q} \subseteq \mathcal{S}$ and $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$ are the minimal sets satisfying the following rules:*

1. *if $s \in \mathcal{I}$, then $s \in \mathcal{Q}$, and*
2. *if $s \in \mathcal{Q}$ and there exists a state–action pair $\langle s, a \rangle \in \pi_F$ such that $s' \in \mathcal{R}(s,a)$, then $s' \in \mathcal{Q}$ and $\mathcal{T}(s, s')$.*

A state $s \in \mathcal{Q}$ is a terminal state *of K if there is no $s' \in \mathcal{Q}$ such that $\mathcal{T}(s, s')$. A* path *of K from $s_0 \in \mathcal{I}$ is a possibly infinite sequence $s_0, s_1, \ldots$ of states in $\mathcal{Q}$ such that, for every state s_i in the sequence, either s_i is the last state of the sequence (in which case s_i is a terminal state of K) or $\mathcal{T}(s_i, s_{i+1})$* holds.

A planning problem for a given domain is described by a set of possible initial states, and a set of goal states.

Definition 5 *Let $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ be a planning domain. A* planning problem *for Σ is a triple $\mathcal{P} = \langle \Sigma, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{I} \subseteq \mathcal{S}$ and $\mathcal{G} \subseteq \mathcal{S}$.*

In the following discussion, $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{V}$, $\mathcal{X}$, $\mathcal{P} = \langle \Sigma, \mathcal{I}, \mathcal{G} \rangle$, π_F, and K represent a planning domain, a finite set of observation variables, an observation function over $\mathcal{S}$ and $\mathcal{V}$, a planning problem for Σ, a state–action table for Σ, and the execution structure induced by π_F from $\mathcal{I}$, respectively.

In many cases, we are interested in *strong solutions* to $\mathcal{P}$, i.e. plans that are guaranteed to achieve the goal in spite of nondeterminism.

Definition 6 *π_F is a* strong solution *to $\mathcal{P}$ if K is acyclic and all terminal states of K are in $\mathcal{G}$.*

In fact, any state–action table π_F for Σ is a strong solution to some planning problems for Σ, if its execution structure K is acyclic and all terminal states of K are in $\mathcal{G}$. So in the following discussion, we use interchangeably the terms *strong solution* and *strong plan*.

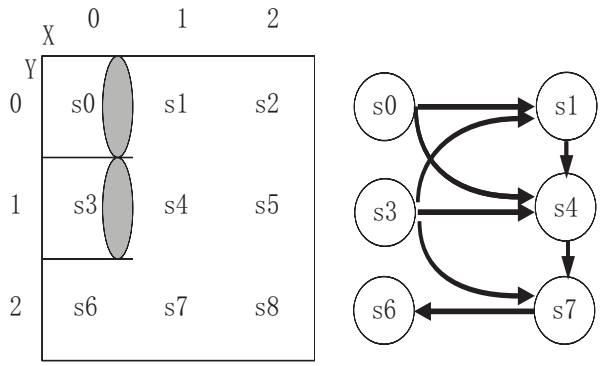

Figure 1: A simple domain and the execution structure induced by π_F from $\mathcal{I}$

In the following example, we illustrate the definitions given in this section.

Example 1 *A simple robot navigation domain Σ is shown on the left of Figure 1. The robot can be in nine positions of a room, corresponding to the states of the domain: $\mathcal{S} = \{s0, \ldots, s8\}$. It can move in the four directions, corresponding to actions $\mathcal{A} =$ {GoNorth, GoSouth, GoEast,GoWest}. An action is applicable only if there is no wall in the direction of motion, e.g. $\mathcal{R}(s0, \text{GoSouth}) = \emptyset$. All the actions are deterministic (e.g., $\mathcal{R}(s7, \text{GoWest}) = \{s6\}$), except for action* GoEast *when on $s0$ (or $s3$), where the robot may slip and end up either in $s1$ or $s4$ (or in $s1$, $s4$, or $s7$), i.e. $\mathcal{R}(s0, \text{GoEast}) = \{s1, s4\}$ and $\mathcal{R}(s3, \text{GoEast}) = \{s1, s4, s7\}$. Let $\mathcal{I} = \{s0, s3\}$ and $\mathcal{G} = \{s6\}$. Then $\pi_F = \{\langle s0, \text{GoEast}\rangle, \langle s1, \text{GoSouth}\rangle, \langle s3, \text{GoEast}\rangle, \langle s4, \text{GoSouth}\rangle, \langle s7, \text{GoWest}\rangle\}$ is a strong plan to $\mathcal{P} = \langle \Sigma, \mathcal{I}, \mathcal{G} \rangle$, and its execution structure K is shown on the right of Figure 1.*

Suppose that the sensors of the robot can detect walls in the current position. This can be formalized with four observation variables, i.e. $WallN$, $WallS$, $WallW$ and $WallE$. In addition, there are six responders (i.e. X_0, X_1, X_2, Y_0, Y_1, and Y_2) located at the north or west border of the room. They work as the X–axis and Y–axis of the nine positions (let $1 \leq n \leq 3$):

1. *The robot sends n pulses north (or west) via the ground.*
2. *If the responder X_{n-1} (or Y_{n-1}) receives n pulses then it sends a responsive pulse south (or east), else it does nothing.*
3. *The robot can detect whether its X-coordinate (or Y-coordinate) is n by receiving a responsive pulse or not in a appointed time.*

This can be formalized with six observation variables, i.e. $X0$, $X1$, $X2$, $Y0$, $Y1$ and $Y2$. So the set of observation variables $\mathcal{V}$ is $\{WallN, WallS, WallW, WallE, X0, X1, X2, Y0, Y1, Y2\}$; and the observation function $\mathcal{X}$ is such that $\mathcal{X}(s0, WallE) = \bot$, $\mathcal{X}(s2, X2) = \top$, and so on.

3 Observation Reduction for Strong Plans

In most real world domains, observation variables are optional and many of them are useless in the execution of a plan (e.g. consider the situation represented in Example 1, the responder $X2$ can be turned off before the execution of π_F); on the other side, information acquisition may require some kind of cost (e.g. time, money, and power etc.). So it is significant to find a minimal set of observation variables which are necessary for the execution of a plan.

In this paper, it is assumed that any two distinct states of the domain can be distinguished by an observation variable at least. So state–action tables are acceptable at execution level. In the execution of a state–action table π_F, the executor is required to observe the values of all the observation variables; consequently, it can know what the current state is and execute the appropriate action at every step. But in many cases, there is much observation information that is useless in the execution of a plan. For example, it makes no sense to observe the value of an observation variable v, where $\mathcal{X}(s, v) = \top$ (or $\mathcal{X}(s, v) = \bot$) for any possible current state s (hereafter, $\mathcal{B}$ denotes the set of possible current states, i.e. the current belief state); and there is no need to do any detection if there exists $a \in \mathcal{A}$ such that $\langle s, a\rangle \in \pi_F$ for any $s \in \mathcal{B}$.

In this section, we define an algorithm that finds an approximate minimal set of observation variables (i.e. $\mathcal{V}_{obs}$) which are necessary for the execution of a strong plan, and transforms the strong plan into a conditional plan branching on the observations built on these observation variables. Before explaining the algorithm in detail, we introduce some notions used in it.

In the execution of π_F, the action that should be executed on s is written as $\textsc{Action}(s, \pi_F)$. That is to say, if $\exists a \in \mathcal{A}.\, \langle s, a\rangle \in \pi_F$ then $\textsc{Action}(s, \pi_F) = a$, else $\textsc{Action}(s, \pi_F) = nop$ (nop means doing nothing).

Let $\emptyset \subset \mathcal{V}_{obs} \subseteq \mathcal{V}$. For each $s \in \mathcal{S}$, we represent the corresponding set of assignments to the observation variables of $\mathcal{V}_{obs}$ with $\textsc{Code}(s, \mathcal{V}_{obs}, \mathcal{X})$, where:

- if $\mathcal{V}_{obs} = \{v\}$ and $\mathcal{X}(s, v) = \top$, then $\textsc{Code}(s, \mathcal{V}_{obs}, \mathcal{X}) = v$.
- if $\mathcal{V}_{obs} = \{v\}$ and $\mathcal{X}(s, v) = \bot$, then $\textsc{Code}(s, \mathcal{V}_{obs}, \mathcal{X}) = \neg v$.
- if $|\mathcal{V}_{obs}| > 1$ and $v \in \mathcal{V}_{obs}$, then $\textsc{Code}(s, \mathcal{V}_{obs}, \mathcal{X}) = \textsc{Code}(s, \{v\}, \mathcal{X}) \wedge \textsc{Code}(s, \mathcal{V}_{obs} - \{v\}, \mathcal{X})$.

The above representation naturally extends to any $\emptyset \subset \mathcal{Q} \subseteq \mathcal{S}$ as follows:

$$\textsc{Code}(\mathcal{Q}, \mathcal{V}_{obs}, \mathcal{X}) = \bigvee_{s \in Q} \textsc{Code}(s, \mathcal{V}_{obs}, \mathcal{X})$$

The set of observations built on $\mathcal{V}$ and $\mathcal{X}$ is $\mathcal{O}(\mathcal{V}, \mathcal{X}) = \{\textsc{Code}(\mathcal{Q}, \mathcal{V}_{obs}, \mathcal{X}) \mid \emptyset \subset \mathcal{Q} \subseteq \mathcal{S}, \emptyset \subset \mathcal{V}_{obs} \subseteq \mathcal{V}\}$.

The definition of conditional plans branching on $\mathcal{O}(\mathcal{V}, \mathcal{X})$ is given as follows. In fact, it is an extension of the one proposed in [Bertoli *et al.*, 2006].

Definition 7 *The set of* conditional plans Π *for* $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$ *is the minimal set such that:*

- $\varepsilon \in \Pi$, *ε is the empty plan;*
- if $a \in \mathcal{A}$ and $\pi \in \Pi$, then $a \circ \pi \in \Pi$;
- *if* $o \in \mathcal{O}(\mathcal{V}, \mathcal{X})$, *and* $\pi_1, \pi_2 \in \Pi$, *then* **if** o **then** π_1 **else** $\pi_2 \in \Pi$.

In general, we are interested in applicable plans, i.e. plans whose execution guarantees that an action is never attempted unless it is applicable, regardless of nondeterminism.

Definition 8 *A plan π for $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$ is* applicable *in state s iff either*

- *π is the empty plan ε; or*
- *π is $a \circ \pi'$, a is applicable in s, and π' is applicable in every $s' \in \mathcal{R}(s, a)$; or*
- *π is* **if** o **then** π_1 **else** π_2, *and:*
  - *if the formula $\textsc{Code}(s, \mathcal{V}, \mathcal{X}) \rightarrow o$ is a tautology, π_1 is applicable in s; and*
  - *if the formula $\textsc{Code}(s, \mathcal{V}, \mathcal{X}) \rightarrow \neg o$ is a tautology, π_2 is applicable in s.*

The execution of a conditional plan is defined in terms of the *runs* associated to it. Intuitively, a run contains the states, observations and actions encountered while executing the plan.

Definition 9 *A* run *is a sequence $\sigma = (s_0, o_0) \circ a_1 \circ (s_1, o_1) \circ \ldots \circ a_n \circ (s_n, o_n)$, where $s_i \in \mathcal{S}$, $o_i \in \mathcal{O}(\mathcal{V}, \mathcal{X})$, and $a_i \in \mathcal{A}$. A sequence σ is a run of π (π is a conditional plan for $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$) from state s iff either*

- *$\pi = \varepsilon$, and $\sigma = (s, o)$ with $o = \textsc{Code}(s, \mathcal{V}, \mathcal{X})$; or*
- *$\pi = a \circ \pi_1$ and $\sigma = (s, o) \circ a \circ \sigma_1$ with $o = \textsc{Code}(s, \mathcal{V}, \mathcal{X})$, and σ_1 is a run for π_1 from some $s_1 \in \mathcal{R}(s, a) \neq \emptyset$; or*
- *$\pi =$* **if** o **then** π_1 **else** π_2, *$\textsc{Code}(s, \mathcal{V}, \mathcal{X}) \rightarrow o$ is a tautology, and σ is a run of π_1 starting from $(s, \textsc{Code}(s, \mathcal{V}, \mathcal{X}))$; or*
- *$\pi =$* **if** o **then** π_1 **else** π_2, *$\textsc{Code}(s, \mathcal{V}, \mathcal{X}) \rightarrow \neg o$ is a tautology, and σ is a run of π_2 starting from $(s, \textsc{Code}(s, \mathcal{V}, \mathcal{X}))$.*

In the following, we write $\textsc{Runs}(s, \pi)$ to denote the set of runs of π starting from s. We write $\textsc{FinalStates}(s, \pi)$ to indicate the set of the final states of the runs of $\textsc{Runs}(s, \pi)$.

The definition of strong plans to $\langle \mathcal{P}, \mathcal{V}, \mathcal{X}\rangle$ is given as follows. In fact, it is similar to the one proposed in [Bertoli *et al.*, 2006].

Definition 10 *A conditional plan π for $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$ is a* strong plan *to $\langle \mathcal{P}, \mathcal{V}, \mathcal{X}\rangle$ iff*

- *π is applicable in every state of $\mathcal{I}$, and*
- *every run of π from a state in $\mathcal{I}$ ends in $\mathcal{G}$, i.e. $\bigcup_{s \in \mathcal{I}} \textsc{FinalStates}(s, \pi) \subseteq \mathcal{G}$.*

Now we can introduce the algorithm for observation reduction of strong plans in detail.

The basic algorithm (i.e. STRONG–FO–PO) is presented in Figure 2. Given a planning problem $\mathcal{P}$, a strong plan π_F to $\mathcal{P}$, a finite set of observation variables $\mathcal{V}$, and an observation function $\mathcal{X}$, the algorithm transforms π_F into a strong plan

```
1.  function STRONG–FO–PO(P, π_F, V, X)
2.    dis ← FINDSTATEPAIRS(P, π_F);
3.    V_obs ← REDUCTION(dis, V, X);
4.    return TRANSFORM(P, π_F, V_obs, X);
5.  end
```

Figure 2: STRONG–FO–PO algorithm

```
1.  function FINDSTATEPAIRS(P, π_F)
      // Suppose P = ⟨Σ, I, G⟩
2.    parts ← DIVIDE(I, π_F);
3.    if (parts = ∅) then
4.      return ∅;
5.    fi;
6.    dis ← ∅;
7.    for each (a, Q) ∈ parts
8.      I_next ← EXECUTE(Σ, Q, a);
9.      P' ← ⟨Σ, I_next, G⟩;
10.     dis ← dis ∪ FINDSTATEPAIRS(P', π);
11.   endfor;
12.   if (I ∩ G ≠ ∅) then
13.     parts ← parts ∪ {(nop, I ∩ G)};
14.   fi;
15.   for any (a_1, Q_1), (a_2, Q_2) ∈ parts,
          where a_1 ≠ a_2
16.     for any s_1 ∈ Q_1, s_2 ∈ Q_2
17.       dis ← dis ∪ {(s_1, s_2)};
18.     endfor;
19.   endfor;
20.   return dis;
21. end
```

Figure 3: FINDSTATEPAIRS procedure

to $\langle \mathcal{P}, \mathcal{V}_{obs}, \mathcal{X} \rangle$, where $\mathcal{V}_{obs}$ is an approximate minimal set of observation variables which are necessary for the execution of π_F. The STRONG–FO–PO algorithm has two main phases: (1) *an observation reduction phase* (lines 2–3), in which $\mathcal{V}_{obs}$ is computed by applications of FINDSTATEPAIRS and REDUCTION, and (2) *a transform phase* (line 4), in which π_F is transformed into a conditional plan for $\langle \Sigma, \mathcal{V}_{obs}, \mathcal{X} \rangle$ by an application of TRANSFORM.

In order to compute $\mathcal{V}_{obs}$, we calculate $dis \subseteq \mathcal{S} \times \mathcal{S}$: if $(s_1, s_2) \in dis$, then $\{s_1, s_2\} \subseteq \mathcal{B}$ (i.e. the set of possible current states) at some step of the execution of π_F and $\text{ACTION}(s_1, \pi_F) \neq \text{ACTION}(s_2, \pi_F)$. This task is accomplished by the FINDSTATEPAIRS procedure, which is presented in Figure 3.

Now let us introduce the FINDSTATEPAIRS procedure in detail:

- First (line 2), the set of possible current states (i.e. $\mathcal{I}$) are divided into some divisions according to the actions specified by π_F. The partition is realized through the function DIVIDE. Formally, $\text{DIVIDE}(\mathcal{I}, \pi_F) = \{(a, \mathcal{Q}) \mid a \in \mathcal{A},\ \emptyset \subset \mathcal{Q} \subseteq \mathcal{I},\ s \in \mathcal{Q} \text{ iff } \langle s, a \rangle \in \pi_F\}$.
- Second (lines 3–5), if $parts = \emptyset$ (i.e. $\mathcal{I} \subseteq \mathcal{G}$), then an empty set is returned by FINDSTATEPAIRS.
- Third (lines 6–11), the elements of dis corresponding to the next belief state are computed by simulating the strong plan's execution for one step and an application of FINDSTATEPAIRS itself. The simulation of a plan is realized through the function EXECUTE. Formally, $\text{EXECUTE}(\Sigma, \mathcal{Q}, a) = \bigcup_{s \in Q} \mathcal{R}(s, a)$.
- Finally (lines 12–19), if the actions specified by π_F for $\mathcal{I}$ are not consistent (i.e. $|parts| > 1$), then any two states (e.g. s_1 and s_2) from different divisions of $\mathcal{I}$ should be distinguished from each other, i.e. (s_1, s_2) or (s_2, s_1) should be included in dis if $a_1 \neq a_2$, $(a_1, \mathcal{Q}_1) \in parts$, $(a_2, \mathcal{Q}_2) \in parts$, $s_1 \in \mathcal{Q}_1$, and $s_2 \in \mathcal{Q}_2$.

```
1.  function REDUCTION(dis, V, X)
      // Suppose V = {v_1, ..., v_n}
2.    for n ≥ i ≥ 1 do di[i] ← 0;
3.    for all (s_1, s_2) ∈ dis
4.      for n ≥ i ≥ 1
5.        if (X(s_1, v_i) ≠ X(s_2, v_i)) then
6.          di[i] ← di[i] + 1;
7.        fi;
8.      endfor;
9.    endfor;
10.   V_obs ← ∅;
11.   while (dis ≠ ∅)
12.     let di[i] = max{di[k] | n ≥ k ≥ 1};
13.     V_obs ← V_obs ∪ {v_i};
14.     for all (s_1, s_2) ∈ dis
15.       if (X(s_1, v_i) ≠ X(s_2, v_i)) then
16.         delete (s_1, s_2) from dis;
17.       fi;
18.     endfor;
19.     di[i] ← 0;
20.   endwhile;
21.   return V_obs;
22. end
```

Figure 4: REDUCTION procedure.

Figure 4 shows the REDUCTION procedure. Given $dis \subseteq \mathcal{S} \times \mathcal{S}$, $\mathcal{V}$, and $\mathcal{X}$ as input, the REDUCTION procedure returns an approximate minimal set of observation variables $\mathcal{V}_{obs}$ such that $\forall (s_1, s_2) \in dis.\, \exists v \in \mathcal{V}_{obs}.\, \mathcal{X}(s_1, v) \neq \mathcal{X}(s_2, v)$. It starts with $\mathcal{V}_{obs} = \emptyset$ and iteratively adds some $v \in \mathcal{V}$ into $\mathcal{V}_{obs}$ until $d = dis$, where $d = \{(s_1, s_2) \in dis \mid \exists v \in \mathcal{V}_{obs}.\, \mathcal{X}(s_1, v) \neq \mathcal{X}(s_2, v)\}$.

The TRANSFORM procedure is presented in Figure 5. Its role is to transform π_F (i.e. a strong plan to $\mathcal{P}$) into a strong plan π to $\langle \mathcal{P}, \mathcal{V}_{obs}, \mathcal{X} \rangle$.

Now let us introduce the TRANSFORM procedure in detail. At first (line 2), the belief state (i.e. $\mathcal{I}$) is divided into some divisions according to the actions specified by π_F. The partition is realized through the function DIVIDE. And then:

1. If $\mathcal{I} \subseteq \mathcal{G}$ (lines 3–4), then an empty plan ε is returned.
2. Else if $\mathcal{I} \cap \mathcal{G} \neq \emptyset$ (lines 5–8), then **if** o **then** ε **else** $\text{TRANSFORM}(\mathcal{P}', \pi_F, \mathcal{V}_{obs}, \mathcal{X})$ is returned, where:

1. **function** TRANSFORM($\mathcal{P}, \pi_F, \mathcal{V}_{obs}, \mathcal{X}$)
   // Suppose $\mathcal{P} = \langle \Sigma, \mathcal{I}, \mathcal{G} \rangle$
2. $parts \leftarrow$ DIVIDE($\mathcal{I}, \pi_F$);
3. **if** $\mathcal{I} \subseteq \mathcal{G}$ **then**
4. **return** ε;
5. **else if** $\mathcal{I} \cap \mathcal{G} \neq \emptyset$ **then**
6. $o \leftarrow$MAKECODE($\mathcal{I} \cap \mathcal{G}, \mathcal{I} - \mathcal{G}, \mathcal{V}_{obs}, \mathcal{X}$);
7. $\mathcal{P}' \leftarrow \langle \Sigma, \mathcal{I} - \mathcal{G}, \mathcal{G} \rangle$;
8. **return if** o **then** ε **else** TRANSFORM($\mathcal{P}', \pi_F, \mathcal{V}_{obs}, \mathcal{X}$);
9. **else if** $parts = \{(a, \mathcal{Q})\}$ **then** // i.e. $|parts| = 1$
10. $\mathcal{I}_{next} \leftarrow$EXECUTE($\Sigma, \mathcal{Q}, a$);
11. $\mathcal{P}' \leftarrow \langle \Sigma, \mathcal{I}_{next}, \mathcal{G} \rangle$;
12. **return** $a \circ$ TRANSFORM($\mathcal{P}', \pi_F, \mathcal{V}_{obs}, \mathcal{X}$);
13. **else**
14. select an element $(a, \mathcal{Q})$ from $parts$;
15. $o \leftarrow$MAKECODE($\mathcal{Q}, \mathcal{I} - \mathcal{Q}, \mathcal{V}_{obs}, \mathcal{X}$);
16. $\mathcal{P}' \leftarrow \langle \Sigma, \mathcal{Q}, \mathcal{G} \rangle$, $\mathcal{P}'' \leftarrow \langle \Sigma, \mathcal{I} - \mathcal{Q}, \mathcal{G} \rangle$;
17. **return if** o **then** TRANSFORM($\mathcal{P}', \pi_F, \mathcal{V}_{obs}, \mathcal{X}$) **else** TRANSFORM($\mathcal{P}'', \pi_F, \mathcal{V}_{obs}, \mathcal{X}$);
18. **fi**;
19. **end**

Figure 5: TRANSFORM procedure.

- o is an observation that can distinguish the states in $\mathcal{I} \cap \mathcal{G}$ from the states in $\mathcal{I} - \mathcal{G}$.
- $\mathcal{P}' = \langle \Sigma, \mathcal{I} - \mathcal{G}, \mathcal{G} \rangle$.

o is computed by MAKECODE($\mathcal{I} \cap \mathcal{G}, \mathcal{I} - \mathcal{G}, \mathcal{V}_{obs}, \mathcal{X}$). Figure 6. shows the MAKECODE subroutine. Given $\mathcal{Q}_1 \subseteq \mathcal{S}$, $\mathcal{Q}_2 \subseteq \mathcal{S}$, $\mathcal{V}_{obs}$, and $\mathcal{X}$ as input, the MAKECODE subroutine finds an approximate minimal set of observation variables $\mathcal{V}_{now} \subseteq \mathcal{V}_{obs}$ such that $\forall (s_1, s_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2. \exists v \in \mathcal{V}_{now}. \mathcal{X}(s_1, v) \neq \mathcal{X}(s_2, v)$, and returns the observation of $\mathcal{Q}_1$ built on $\mathcal{V}_{now}$ and $\mathcal{X}$ (i.e. CODE($\mathcal{Q}_1, \mathcal{V}_{now}, \mathcal{X}$)).

3. Else if the actions specified by π_F for $\mathcal{I}$ are consistent, i.e. $|parts| = 1$ (lines 9–12), then $a \circ$ TRANSFORM($\mathcal{P}', \pi_F, \mathcal{V}_{obs}, \mathcal{X}$) is returned, where:
   - a is the action specified by π_F for all the possible current states.
   - $\mathcal{P}' = \langle \Sigma, \mathcal{I}_{next}, \mathcal{G} \rangle$. $\mathcal{I}_{next}$ is the belief state at next step, and it is computed by EXECUTE($\Sigma, \mathcal{Q}, a$).
4. Else (lines 13–17) $(a, \mathcal{Q})$ is selected from $parts$, and **if** o **then** π_1 **else** π_2 is returned, where:
   - o is an observation that can distinguish the states in $\mathcal{Q}$ from the states in $\mathcal{I} - \mathcal{Q}$, and it is computed by MAKECODE($\mathcal{Q}, \mathcal{I} - \mathcal{Q}, \mathcal{V}_{obs}, \mathcal{X}$).
   - $\pi_1 =$TRANSFORM($\langle \Sigma, \mathcal{Q}, \mathcal{G} \rangle, \pi_F, \mathcal{V}_{obs}, \mathcal{X}$).
   - $\pi_2 =$TRANSFORM($\langle \Sigma, \mathcal{I} - \mathcal{Q}, \mathcal{G} \rangle, \pi_F, \mathcal{V}_{obs}, \mathcal{X}$).

From Definition 6, Definition 10 and the algorithms presented above, we can get Theorem 1 and Theorem 2.

Theorem 1 *If π_F is a strong plan to $\mathcal{P}$, then* STRONG–FO–PO($\mathcal{P}, \pi_F, \mathcal{V}, \mathcal{X}$) *terminates.*

1. **function** MAKECODE($\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{V}_{obs}, \mathcal{X}$)
2. $dis \leftarrow \emptyset$;
3. **for** any $s_1 \in \mathcal{Q}_1, s_2 \in \mathcal{Q}_2$
4. $dis \leftarrow dis \cup \{(s_1, s_2)\}$;
5. **endfor**;
6. $\mathcal{V}_{now} \leftarrow$ REDUCTION($dis, \mathcal{V}_{obs}, \mathcal{X}$);
7. **return** CODE($\mathcal{Q}_1, \mathcal{V}_{now}, \mathcal{X}$);
8. **end**

Figure 6: MAKECODE subroutine.

In fact, Theorem 1 can be proved by showing that all the paths of the execution structure K induced by π_F from $\mathcal{I}$ are finite. According to Definition 6, all the paths of K are finite (i.e. K is acyclic) because π_F is a strong plan to $\mathcal{P}$.

Theorem 2 *Let π_F be a strong plan to $\mathcal{P}$. Let $\mathcal{Q}_t$ be the set of terminal states of K (i.e. the execution structure induced by π_F from $\mathcal{I}$). Suppose π is the conditional plan returned by* STRONG–FO–PO($\mathcal{P}, \pi_F, \mathcal{V}, \mathcal{X}$). *Then:*

- *π is a strong plan to $\langle \mathcal{P}, \mathcal{V}, \mathcal{X} \rangle$, and*
- $\bigcup_{s \in \mathcal{I}}$ FINALSTATES(s, π) $= \mathcal{Q}_t \subseteq \mathcal{G}$.

In the following example, we illustrate the algorithms described in this section.

Example 2 *Consider the situation depicted in Example 1. We apply* STRONG–FO–PO *to* ($\mathcal{P}, \pi_F, \mathcal{V}, \mathcal{X}$). *Firstly, dis is computed by* FINDSTATEPAIRS($\mathcal{P}, \pi_F$), *and it is $\{(s1, s7), (s4, s7)\}$; and then, $\mathcal{V}_{obs}$ is computed by* REDUCTION($dis, \mathcal{V}, \mathcal{X}$), *and it is $\{WallS\}$; lastly, π is returned by* TRANSFORM($\mathcal{P}, \pi_F, \mathcal{V}_{obs}, \mathcal{X}$), *where:*

- $\pi = \pi_0 =$ GoEast $\circ\ \pi_1$, *and*
- $\pi_1 =$ **if** $WallS$ **then** (GoWest $\circ\ \varepsilon$) **else** π_2, *and*
- $\pi_2 =$ GoSouth $\circ\ \pi_3$, *and*
- $\pi_3 =$ **if** $WallS$ **then** (GoWest $\circ\ \varepsilon$) **else** π_4, *and*
- $\pi_4 =$ GoSouth $\circ$ GoWest $\circ \varepsilon$.

It is easy to find that π is a strong plan to $\langle \mathcal{P}, \mathcal{V}, \mathcal{X} \rangle$, and $\mathcal{G} = \bigcup_{s \in \mathcal{I}}$ FINALSTATES(s, π) $= \{s6\}$ (i.e. the set of terminal states of the execution structure induced by π_F from $\mathcal{I}$).

In fact, the observation variables of $\mathcal{V} - \mathcal{V}_{obs}$ are useless in the whole executions of π and π_F. So the sensors and the responders that correspond to these observation variables (i.e. $WallN$, $WallW$, $WallE$, $X0$, $X1$, $X2$, $Y0$, $Y1$, and $Y2$) can be turned off before executing π. Furthermore, it is not necessary to observe all the values of observation variables of $\mathcal{V}_{obs}$ at every step: for instance, π requires the plan executor to execute GoEast *at first, regardless of the observation information about the current state.*

4 Conclusions

This paper presents an algorithm (i.e. STRONG–FO–PO) for observation reduction of strong plans under full observability, which take the form of state–action tables. To the best of our knowledge, no other work has discussed and tackled this problem.

Given a planning problem $\mathcal{P} = \langle \Sigma, \mathcal{I}, \mathcal{G} \rangle$, a strong plan π_F to $\mathcal{P}$, a finite set of observation variables $\mathcal{V}$, and an observation function $\mathcal{X}$ as input, the STRONG–FO–PO algorithm finds $\mathcal{V}_{obs}$, i.e. an approximate minimal set of observation variables which are necessary for the execution of π_F, and transforms π_F into a conditional plan π branching on the observations built on $\mathcal{V}_{obs}$ and $\mathcal{X}$, such that:

- π is a strong plan to $\langle \mathcal{P}, \mathcal{V}_{obs}, \mathcal{X} \rangle$, and
- $\bigcup_{s \in \mathcal{I}} \text{FINALSTATES}(s, \pi) = \mathcal{Q}_t \subseteq \mathcal{G}$, where $\mathcal{Q}_t$ is the set of terminal states of the execution structure induced by π_F from $\mathcal{I}$.

The STRONG–FO–PO algorithm always terminates.

In the STRONG–FO–PO algorithm, there are two levels of the meaning of observation reduction:

1. The observation variables which are useless in the whole execution of a plan π_F, are discarded as more as possible. That is to say, the algorithm computes $\mathcal{V}_{obs}$ and discards the observation variables of $\mathcal{V} - \mathcal{V}_{obs}$.
2. At every step of the execution of π_F, values of the observation variables that are useless in choosing the appropriate action, are discarded as more as possible. That is to say, the algorithm computes $\mathcal{V}_{now}$ (see Figure 6, line 6) for every step of the execution of π_F, and discards the values of observation variables of $\mathcal{V}_{obs} - \mathcal{V}_{now}$ temporarily.

In the future, we plan to extend this work along the following directions.

- We will investigate the problem under the situation that:
  - observation variables may relate to different data types (symbolic, numeric, ordinal, etc.), and
  - noisy observations or missing values may be allowed.
- Considering the costs corresponding to observation variables may be different, we will investigate the problem of observation reduction for some plans, in which some preferences should be taken into account.
- We will investigate observation reduction for plans to other kinds of planning problems, in particular strong cyclic planning problems [Cimatti *et al.*, 1998a; Daniele *et al.*, 1999], and planing problems for temporally extended goals [Pistore and Traverso, 2001; Lago *et al.*, 2002].

References

[Bertoli *et al.*, 2001] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 473–478, Seattle, WA, USA, 2001. Morgan Kaufmann.

[Bertoli *et al.*, 2006] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Strong planning under partial observability. *Artificial Intelligence*, 170:337–384, 2006.

[Bonet and Geffner, 2000] Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the 5th International Conference on AI Planning Systems*, pages 52–61, Breckenridge, Colorado, USA, 2000. AAAI Press.

[Boutilier, 2002] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 239–246, Edmonton, Canada, 2002. AAAI Press.

[Cimatti *et al.*, 1998a] Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 875–881, Madison, Wisconsin, USA, 1998. AAAI Press.

[Cimatti *et al.*, 1998b] Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Strong planning in non–deterministic domains via model checking. In *Proceedings of the 4th International Conference on AI Planning Systems*, pages 36–43, Pittsburgh, USA, 1998. AAAI Press.

[Cimatti *et al.*, 2003] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147:35–84, 2003.

[Daniele *et al.*, 1999] Marco Daniele, Paolo Traverso, and Moshe Y. Vardi. Strong cyclic planning revisited. In *Proceedings of the 5th European Conference on Planning*, pages 35–48, Durham, United Kingdom, 1999. Springer-Verlag.

[Eiter *et al.*, 2003] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. A logic programming approach to knowledge–state planning, II: The DLV^K system. *Artificial Intelligence*, 144(1–2):157–211, 2003.

[Herzig *et al.*, 2003] Andreas Herzig, Jerome Lang, and Pierre Marquis. Action representation and partially observable planning using epistemic logic. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence 2003*, pages 1067–1072, Acapulco, Mexico, 2003. Morgan Kaufmann.

[Lago *et al.*, 2002] Ugo Dal Lago, Marco Pistore, and Paolo Traverso. Planning with a language for extended goals. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 447–454, Edmonton, Canada, 2002. AAAI Press.

[Pistore and Traverso, 2001] Marco Pistore and Paolo Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 479–484, Seattle, WA, USA, 2001. Morgan Kaufmann.