Structured Plans and Observation Reduction for Plans with Contexts

Wei Huang[1], **Zhonghua Wen**[2], **Yunfei Jiang**[3], **Hong Peng**[1]
1. School of Computer Science & Engineering,
South China University of Technology, Guangzhou, China
2. College of Information Engineering, Xiangtan University, Xiangtan, China
3. Software Research Institute, School of Information Science and Technology,
Sun Yat-sen University, Guangzhou, China
Email: huangbodao@yahoo.com.cn

Abstract

In many real world planning domains, some observation information is optional and useless to the execution of a plan; on the other hand, information acquisition may require some kind of cost. The problem of observation reduction for strong plans has been addressed in the literature. However, observation reduction for plans with contexts (which are more general and useful than strong plans in robotics) is still a open problem. In this paper, we present an attempt to solve the problem. Our first contribution is the definition of structured plans, which can encode sequential, conditional and iterative behaviors, and is expressive enough for dealing with incomplete observation information and internal states of the agent. A second contribution is an observation reduction algorithm for plans with contexts, which can transform a plan with contexts into a structured plan that only branches on necessary observation information.

1 Introduction

Planning for extended goals in nondeterministic domains [Pistore and Traverso, 2001; Lago *et al.*, 2002], is a well known and significant problem in AI. In nondeterministic and full observable domains, actions are modelled with different outcomes that cannot be predicted at planning time, and the state of the world can be completely identified via some possible observation at execution time. Therefore, it is the job of plans to specify what to do depending on the observation information gathered at execution time, and the context of execution, i.e., the internal state of the agent.

The plan synthesis problem has been addressed in the literature (e.g., see [Pistore and Traverso, 2001; Lago *et al.*, 2002]), where plans are expressed as policies on state–context pairs (i.e., plans with contexts). The agent executing plans with contexts, is considered able to sense the state of the world directly, and depending on the state of the world and the internal state of the agent, execute different actions. In most real world domains, however, the agent distinguishes states of the world by acquiring some optional observation information at some cost (battery, money, and time, etc.). On the other hand, some observation information may be unnecessary to the plan execution. So it is significant to reduce the unnecessary observation tasks for plans with contexts. A simpler subset of this problem, i.e., observation reduction for strong plans [Cimatti *et al.*, 2003] has been already addressed in [Huang *et al.*, 2007], but to the best of our knowledge, observation reduction for general plans with contexts is still an open problem. In this paper, we present a first attempt to solve the problem.

At first, we extend conditional plans [Huang *et al.*, 2007; Bertoli *et al.*, 2006] to structured plans that can encode sequential, conditional and iterative behaviors, and are expressive enough for dealing with incomplete observation information and with internal states of the agent. Intuitively, structured plans are similar to programs with conditional statements and loops, in which contexts are used to represent the destination points of jumps occurring in execution control. We also define what does "a structured plan is equivalent to a plan with contexts" mean, and the expected cost of observation information acquisition required at every step of the execution of a structured plan.

And then, we define an observation reduction algorithm for general plans with contexts. The algorithm can transform a plan with contexts into a structured plan that only branches on necessary observation information. The algorithm terminates, and is correct (i.e., the structured plan returned by the algorithm is equivalent to the plan with contexts).

This paper is structured as follows. Firstly, we review some basic notions that are relevant to our work (see [Pistore and Traverso, 2001; Huang *et al.*, 2007] for further details and examples). Secondly, we introduce some definitions about structured plans and their execution. Thirdly, we describe an observation reduction algorithm for plans with contexts, and show some properties of the algorithm. Finally, we draw some conclusions and discuss future research directions.

2 Preliminaries

A planning domain $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{R} \rangle$ is a model of a generic system with its own dynamics, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{I} \subseteq \mathcal{S}$ is the set of initial states (we require $\mathcal{I} \neq \emptyset$), and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$ is the state–transition function. $\textsc{Act}(s) = \{a \in \mathcal{A} | \exists s' \in \mathcal{R}(s, a)\}$ denotes the set of actions that are applicable in $s \in \mathcal{S}$.

A plan with contexts π_C for Σ is a tuple $\langle \mathcal{C}, c_0, act, ctxt \rangle$ where: $\mathcal{C}$ is a set of contexts, $c_0 \in \mathcal{C}$ is the initial context, $act : \mathcal{S} \times \mathcal{C} \rightarrow \mathcal{A}$ is the action function, and $ctxt : \mathcal{S} \times \mathcal{C} \times \mathcal{S} \rightarrow \mathcal{C}$ is the context function. The execution of π_C on Σ can be described in terms of transitions between state–context pairs: $\langle s, c \rangle \overrightarrow{a} \langle s', c' \rangle$ if $s' \in \mathcal{R}(s, a)$, $a = act(s, c)$, and $c' = ctxt(s, c, s')$. A run of plan π_C from state s_0 is a possibly infinite sequence $\langle s_0, c_0 \rangle \overrightarrow{a_0} \langle s_1, c_1 \rangle \overrightarrow{a_1} \ldots$ where $\langle s_i, c_i \rangle \overrightarrow{a_i} \langle s_{i+1}, c_{i+1} \rangle$ are transitions. The execution structure $K = \langle \mathcal{SC}, \mathcal{SC}_0, \mathcal{TR} \rangle$ is the finite presentation of the set of all possible runs of π_C on Σ, where:

- $\mathcal{SC}_0 = \mathcal{I} \times \{c_0\}$;
- $\mathcal{SC}$ is the minimal set satisfying the following rules:
  - $\mathcal{SC}_0 \subseteq \mathcal{SC}$,
  - if $\langle s, c \rangle \in \mathcal{SC}$, and there exists $a \in \mathcal{A}$ and $\langle s', c' \rangle \in \mathcal{S} \times \mathcal{C}$ such that $\langle s, c \rangle \overrightarrow{a} \langle s', c' \rangle$, then $\langle s', c' \rangle \in \mathcal{SC}$;
- and $\langle \langle s, c \rangle, \langle s', c' \rangle \rangle \in \mathcal{TR}$ if $\langle s, c \rangle \overrightarrow{a} \langle s', c' \rangle$ for some a.

The agent executing plans with contexts, is considered able to sense the state of the world directly, and depending on the state of the world and the internal state of the agent (i.e., the context), execute different actions. In practice, however, the agent distinguishes states of the world by acquiring some optional observation information at some cost (e.g., time, money, and power etc.). An observation setting can be modelled by a set of observation variables $\mathcal{V}$, and an observation function $\mathcal{X} : \mathcal{S} \times \mathcal{V} \rightarrow \{\top, \bot\}$.[1] The observation of a set of states $\emptyset \subset \mathcal{S}' \subseteq \mathcal{S}$ built on $\emptyset \subset \mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{X}$ can be expressed as a propositional formula, and it is:

$$\textsc{Obs}(\mathcal{S}', \mathcal{V}', \mathcal{X}) = \bigvee_{s \in \mathcal{S}'} \Big(\bigwedge_{v \in \mathcal{V}'} \textsc{Code}(s, v, \mathcal{X}) \Big), where$$

$$\textsc{Code}(s, v, \mathcal{X}) = \begin{cases} v & \text{iff } \mathcal{X}(s, v) = \top \\ \neg v & \text{otherwise} \end{cases}$$

$\mathcal{O}(\mathcal{S}, \mathcal{V}, \mathcal{X}) = \{\textsc{Obs}(\mathcal{S}', \mathcal{V}', \mathcal{X}) \mid \emptyset \subset \mathcal{S}' \subseteq \mathcal{S}, \emptyset \subset \mathcal{V}' \subseteq \mathcal{V}\}$ is the set of all possible observations built from $\langle \mathcal{S}, \mathcal{V}, \mathcal{X} \rangle$. Furthermore, for each observation variable $v \in \mathcal{V}$, we use an integer $\textsc{Cost}(v)$ to represent the cost of each time sensing the value of v.

In the following example[2], that will be used throughout the paper, we will illustrate the notions introduced in this section.

Example 1 *The left of Figure 1 shows a simple robot navigation domain Σ which contains a 3×3 room. The states of Σ (i.e., $\mathcal{S} = \{s_0, \ldots, s_8\}$) correspond to the 9 positions in the room. s_0 and s_1 are the possible initial states of Σ, i.e., $\mathcal{I} = \{s_0, s_1\}$. The robot can move in the four directions (i.e., $\mathcal{A} = \{\rightarrow, \downarrow, \leftarrow, \uparrow\}$). An action is applicable only if there is no wall in the direction of motion. All the actions are deterministic (e.g., $\mathcal{R}(s_3, \downarrow) = \{s_4\}$), except for action $\rightarrow$:*

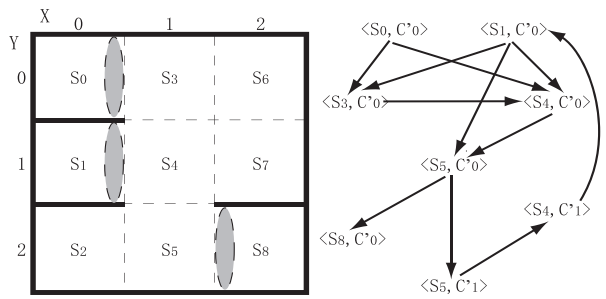

Figure 1: A simple domain Σ and the execution structure corresponding to π_C and Σ

s	c	$act(s, c)$	$s' \in \mathcal{R}(s, a)$	$ctxt(s, c, s')$
s_0	c'_0	$\rightarrow$	s_3 or s_4	c'_0
s_1	c'_0	$\rightarrow$	s_3, s_4, or s_5	c'_0
s_3	c'_0	$\downarrow$	s_4	c'_0
s_4	c'_0	$\downarrow$	s_5	c'_0
s_5	c'_0	$\rightarrow$	s_8	c'_0
s_5	c'_0	$\rightarrow$	s_5	c'_1
s_5	c'_1	$\uparrow$	s_4	c'_1
s_4	c'_1	$\leftarrow$	s_1	c'_0

Table 1: A plan with contexts π_C for Σ

- *when on s_0 (or s_1), the robot may slip and end up either in s_3 or s_4 (or in s_3, s_4, or s_5), i.e., $\mathcal{R}(s_0, \rightarrow) = \{s_3, s_4\}$ and $\mathcal{R}(s_1, \rightarrow) = \{s_3, s_4, s_5\}$;*
- *when on s_5, the robot may be blocked by a door and end up either in s_5 or s_8, i.e., $\mathcal{R}(s_5, \rightarrow) = \{s_5, s_8\}$.*

Let $\pi_C = \langle \mathcal{C}', c'_0, act, ctxt \rangle$ be the plan with contexts shown in Table 1. The execution structure K corresponding to π_C and Σ is shown on the right of Figure 1.

Suppose the robot can detect the position of the walls neighboring its location, and perceive whether the X-coordinate (or Y-coordinate) of its location is $n (0 \leq n \leq 2)$ (i.e., the set of observation variables $\mathcal{V} = \{$E,S,W,N,X0,X1,X2,Y0,Y1,Y2$\}$). So the observation function $\mathcal{X}$ is such that $\mathcal{X}(s_0,$ E$) = \bot$, $\mathcal{X}(s_5,$ X1$) = \top$, and so on. $\textsc{Obs}(\mathcal{I}, \{E,S\}, \mathcal{X}) = \neg$E$ \wedge$ S. *For the sake of simplicity, we assume that $\textsc{Cost}(v) = 1$ for each $v \in \mathcal{V}$. The agent executing π_C, has to observe all the observation variables before it selects an action to execute. Hence, the total cost of observation information acquisition required at every step of the execution of π_C is $\sum_{v \in \mathcal{V}} \textsc{Cost}(v) = |\mathcal{V}| = 10$.*

In the rest of this paper, $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{R} \rangle$, $\mathcal{V}$, and $\mathcal{X}$ represent a planning domain, a finite set of observation variables, and an observation function over $\mathcal{S}$ and $\mathcal{V}$, respectively.

3 Structured Plans

In this section, we present the definition of structured plans, which can encode sequential, conditional and iterative behaviors, and is expressive enough for dealing with incomplete observation information and with internal states of the agent.

[1] In fact, $\langle \mathcal{V}, \mathcal{X} \rangle$ can be used to model partial observability (e.g., [Bertoli *et al.*, 2001; 2006]). However, for the sake of executability of plans with contexts, we assume that any two distinct states of the world can be distinguished by one observation variable at least.

[2] The planning domain and the observation setting introduced in this example are modified from those given in [Huang *et al.*, 2007].

Intuitively, structured plans are similar to programs with conditional statements and loops. An empty plan ε means doing nothing except sending a signal of termination. Contexts work as destination points of jumps that take place in execution control, and the initial context represents the place where the whole execution begins. For each context, the explanation function specifies the rest of the plan to be executed from the point corresponding to the context.

Definition 1 *A* structured plan *for $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ is a tuple $\pi_S = \langle \mathcal{C}, c_0, \mathcal{E} \rangle$, where $\mathcal{C}$ is a finite set of contexts, $c_0 \in \mathcal{C}$ is the initial context, and $\mathcal{E} : \mathcal{C} \to \Pi$ is the explanation function. Π is the set of sub–plans built on $\langle \Sigma, \mathcal{V}, \mathcal{X}, \mathcal{C} \rangle$, and it is the minimal set satisfying the following rules:*

- $\mathcal{C} \cup \{\varepsilon\} \subseteq \Pi$, *$\varepsilon$ is the empty plan;*
- *if $a \in \mathcal{A}$ and $\pi \in \Pi$, then $a \circ \pi \in \Pi$;*
- *if $\emptyset \subset \mathcal{V}' \subseteq \mathcal{V}$, and $\langle o_i, \pi_i \rangle \in \mathcal{O}(\mathcal{S}, \mathcal{V}', \mathcal{X}) \times \Pi$ for all $1 \leq i \leq n$, then* **switch**$(\mathcal{V}')\{\langle o_i, \pi_i \rangle | 1 \leq i \leq n\} \in \Pi$.

In the following discussion, $\pi_S = \langle \mathcal{C}, c_0, \mathcal{E} \rangle$ represents a structured plan for $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$. We say that π_S is *simple*, if $\mathcal{C} \cap \{\mathcal{E}(c) | c \in \mathcal{C}\} = \emptyset$. Intuitively, successive jumps are not allowed in the execution control of a simple structured plan. In the rest of the paper, we consider only structured plans that are simple. If $o, o' \in \mathcal{O}(\mathcal{S}, \mathcal{V}, \mathcal{X})$ and $o \to o'$ is a tautology, then we write $o \Rightarrow o'$.

The execution of π_S is defined in terms of the *runs* associated to it. Intuitively, a run σ is a sequence of configurations [Sardina *et al.*, 2006] that, describe the world state, the belief state [Bonet and Geffner, 2000], and the rest of the plan to be executed.

Definition 2 *A* configuration *for $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and structured plan π_S is a tuple $\langle s, \mathcal{B}, \pi \rangle \in \mathcal{S} \times 2^{\mathcal{S}} \times \Pi$.*

Configuration $\langle s, \mathcal{I}, \mathcal{E}(c_0) \rangle$ is initial if $s \in \mathcal{I}$. Configuration $\langle s, \mathcal{B}, \pi \rangle$ may evolve into configuration $\langle s', \mathcal{B}', \pi' \rangle$, written $\langle s, \mathcal{B}, \pi \rangle \longrightarrow \langle s', \mathcal{B}', \pi' \rangle$, if either:

- *$\pi = a \circ \pi''$, if $\pi'' \notin \mathcal{C}$ then $\pi' = \pi''$ else $\pi' = \mathcal{E}(\pi'')$, $s' \in \mathcal{R}(s, a)$ and $\mathcal{B}' = \cup_{s'' \in \mathcal{B}} \mathcal{R}(s'', a)$; or*
- *$\pi =$* **switch**$(\mathcal{V}')\{\langle o_1, \pi_1 \rangle, \ldots, \langle o_n, \pi_n \rangle\}$, $\textsc{Obs}(\{s\}, \mathcal{V}', \mathcal{X}) \Rightarrow o_i$, *if $\pi_i \notin \mathcal{C}$ then $\pi' = \pi_i$ else $\pi' = \mathcal{E}(\pi_i)$, $s' = s$, and $\mathcal{B}' = \{s'' \in \mathcal{B} | \textsc{Obs}(\{s''\}, \mathcal{V}', \mathcal{X}) \Rightarrow o_i\}$.*

The reachable configurations for $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and π_S, are defined by the following rules:

- *if $\langle s, \mathcal{B}, \pi \rangle$ is initial, then it is reachable;*
- *if $\langle s, \mathcal{B}, \pi \rangle$ is reachable and $\langle s, \mathcal{B}, \pi \rangle \longrightarrow \langle s', \mathcal{B}', \pi' \rangle$, then $\langle s', \mathcal{B}', \pi' \rangle$ is also reachable.*

Due to the nondeterminism in a planning domain, there may be many different runs of a structured plan. We use an execution structure to represent the set of all possible runs.

Definition 3 *The* execution structure *corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and structured plan π_S, is $K = \langle \mathcal{Q}, \mathcal{Q}_0, \mathcal{T} \rangle$, where: $\mathcal{Q}$ is the set of reachable configurations; $\mathcal{Q}_0 = \{\langle s, \mathcal{I}, \mathcal{E}(c_0) \rangle | s \in \mathcal{I}\}$ is the set of initial configurations; and $\mathcal{T} = \{\langle q, q' \rangle \in \mathcal{Q} \times \mathcal{Q} | q \longrightarrow q'\}$.*

In the following discussion, $K = \langle \mathcal{Q}, \mathcal{Q}_0, \mathcal{T} \rangle$ represents the execution structure corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and π_S. $\textsc{Terc}(K) = \{q \in \mathcal{Q} | \forall q' \in \mathcal{Q}. \langle q, q' \rangle \notin \mathcal{T}\}$ denotes the set of terminal configurations of execution structure K. A run of K from q_0 is a possibly infinite sequence $(q_0, q_1, \ldots)$ such that, for every q_i, either $q_i \in \textsc{Terc}(K)$ or $\langle q_i, q_{i+1} \rangle \in \mathcal{T}$. $\textsc{Runs}(q, K)$ denotes the set of runs of K from q.

In general, we are interested in executable structured plans, i.e., structured plans whose execution guarantees that an action is never attempted unless it is applicable, and that for any possible observation information, they say what to do next without ambiguity.

Definition 4 *Let $K = \langle \mathcal{Q}, \mathcal{Q}_0, \mathcal{T} \rangle$ be the execution structure corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and structured plan π_S. π_S is* executable *on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ if:*

- *for all $\langle s, \mathcal{B}, \pi \rangle \in \textsc{Terc}(K)$, π is ε; and*
- *for all $\langle s, \mathcal{B}, \pi \rangle \in \mathcal{Q}$ such that π takes the form of* **switch**$(\mathcal{V}')\{\langle o_i, \pi_i \rangle | 1 \leq i \leq n\}$, *there exists only one $1 \leq i \leq n$ such that $\textsc{Obs}(\{s\}, \mathcal{V}', \mathcal{X}) \Rightarrow o_i$.*

Sometimes we may only be interested in the state transitions induced by (the execution of) a structured plan. Let $q_0 = \langle s_0, \mathcal{B}_0, \pi_0 \rangle \in \mathcal{Q}$, and $\sigma = (\langle s_0, \mathcal{B}_0, \pi_0 \rangle, \langle s_1, \mathcal{B}_1, \pi_1 \rangle, \ldots, \langle s_i, \mathcal{B}_i, \pi_i \rangle, \ldots) \in \textsc{Runs}(q_0, K)$, then we use $\textsc{Strs}(\sigma) = s'_0 \diamond a_1 \diamond s'_1 \diamond \ldots$ (where $s'_i \in \mathcal{S}$, $a_{i+1} \in \mathcal{A}$) to represent the state transitions corresponding to σ such that:

- if $\pi_0 = \varepsilon$, then $\textsc{Strs}(\sigma) = s_0$; or else
- if $\pi_0 = a \circ \ldots$ such that $a \in \mathcal{A}$, then $\textsc{Strs}(\sigma) = s_0 \diamond a \diamond s'_1 \diamond \ldots$ such that $s'_1 \diamond \ldots = \textsc{Strs}((\langle s_1, \mathcal{B}_1, \pi_1 \rangle, \ldots, \langle s_i, \mathcal{B}_i, \pi_i \rangle, \ldots))$; or else
- $\textsc{Strs}(\sigma) = \textsc{Strs}((\langle s_1, \mathcal{B}_1, \pi_1 \rangle, \ldots, \langle s_i, \mathcal{B}_i, \pi_i \rangle, \ldots))$.

Now we can compare the behaviors induced by (the execution of) a structured plan π_S with those induced by a plan with contexts π_C.

Definition 5 *Let $\pi_C = \langle \mathcal{C}', c'_0, act, ctxt \rangle$ be a plan with contexts for Σ. Then π_S is* equivalent *to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ iff:*

- *for each $s_0 \in \mathcal{I}$ and $\sigma \in \textsc{Runs}(\langle s_0, \mathcal{I}, \mathcal{E}(c_0) \rangle, K)$ such that $\textsc{Strs}(\sigma) = s_0 \diamond a_1 \diamond s_1 \diamond a_2 \diamond \ldots$, there exists a run $\sigma' = \langle s_0, c'_0 \rangle \overrightarrow{a_1} \langle s_1, c'_1 \rangle \overrightarrow{a_2} \ldots$ of π_C such that $c'_i \in \mathcal{C}'$;*
- *and for each $s_0 \in \mathcal{I}$ and run σ' of π_C from s_0 (let $\sigma' = \langle s_0, c'_0 \rangle \overrightarrow{a_1} \langle s_1, c'_1 \rangle \overrightarrow{a_2} \ldots$), there exists $\sigma \in \textsc{Runs}(\langle s_0, \mathcal{I}, \mathcal{E}(c_0) \rangle, K)$ such that $\textsc{Strs}(\sigma) = s_0 \diamond a_1 \diamond s_1 \diamond a_2 \diamond \ldots$*

Definition 5 only concerns the state transitions induced by a structured plan and a plan with contexts. However, in many real world domains, the cost of observation information acquisition is an issue.

Definition 6 *Let $\langle s_0, \mathcal{B}_0, \pi_0 \rangle \in \mathcal{Q}$ and $\sigma \in \textsc{Runs}(\langle s_0, \mathcal{B}_0, \pi_0 \rangle, K)$ such that $\sigma = (\langle s_0, \mathcal{B}_0, \pi_0 \rangle, \langle s_1, \mathcal{B}_1, \pi_1 \rangle, \ldots, \langle s_i, \mathcal{B}_i, \pi_i \rangle, \ldots)$. Then the* expected cost *of observation information acquisition required at every step of the execution (i.e., choosing an action) corresponding to σ is:*

$$\textsc{AvOC}(\sigma) = \frac{\sum_{i \geq 0} \textsc{OC}(\pi_i)}{1 + \sum_{i \geq 0} \textsc{NoA}(\pi_i)}, \text{ where}$$

$$\textsc{OC}(\pi_i) = \begin{cases} \sum_{v \in \mathcal{V}'} \textsc{Cost}(v) & \text{iff } \pi_i = \textbf{switch}(\mathcal{V}')\{\ldots\} \\ 0 & \text{otherwise} \end{cases}$$

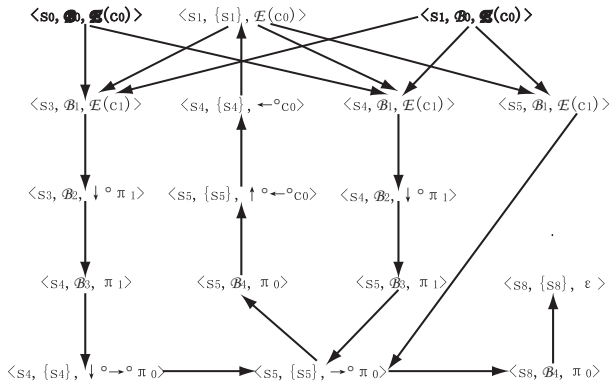

Figure 2: The execution structure corresponding to π_S

$$\text{NOA}(\pi_i) = \begin{cases} 1 & \text{iff } \pi_i = a \circ \dots \\ 0 & \text{otherwise} \end{cases}$$

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

Example 2 *Consider the situation depicted in Example 1. Let $\pi_S = \langle \mathcal{C}, c_0, \mathcal{E} \rangle$ be a structured plan for $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$, where:*

- $\mathcal{C} = \{c_0, c_1\}$, $\mathcal{E}(c_0) = \rightarrow \circ c_1$,
- $\mathcal{E}(c_1) =$ **switch**$(\{\mathsf{S}\})\{\langle \mathsf{S}, \rightarrow \circ \pi_0 \rangle, \langle \neg\mathsf{S}, \downarrow \circ \pi_1 \rangle\}$,
- $\pi_0 =$ **switch**$(\{\mathsf{E}\})\{\langle \mathsf{E}, \varepsilon \rangle, \langle \neg\mathsf{E}, \uparrow \circ \leftarrow \circ c_0 \rangle\}$, and
- $\pi_1 =$ **switch**$(\{\mathsf{S}\})\{\langle \mathsf{S}, \rightarrow \circ \pi_0 \rangle, \langle \neg\mathsf{S}, \downarrow \circ \rightarrow \circ \pi_0 \rangle\}$.

Let $\mathcal{B}_0 = \mathcal{I}$, $\mathcal{B}_1 = \{s_3, s_4, s_5\}$, $\mathcal{B}_2 = \{s_3, s_4\}$, $\mathcal{B}_3 = \{s_4, s_5\}$, and $\mathcal{B}_4 = \{s_5, s_8\}$. The execution structure $K = \langle \mathcal{Q}, \mathcal{Q}_0, \mathcal{T} \rangle$ corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and π_S is shown (as a directed graph) in Figure 2. It is easy to find that π_S is simple and executable on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$.

In Figure 2, we can find that $\sigma_1 = (\langle s_1, \mathcal{B}_0, \mathcal{E}(c_0) \rangle, \langle s_5, \mathcal{B}_1, \mathcal{E}(c_1) \rangle, \langle s_5, \{s_5\}, \rightarrow \circ \pi_0 \rangle, \langle s_5, \mathcal{B}_4, \pi_0 \rangle, \langle s_5, \{s_5\}, \uparrow \circ \leftarrow \circ c_0 \rangle, \langle s_4, \{s_4\}, \leftarrow \circ c_0 \rangle, \langle s_1, \{s_1\}, \mathcal{E}(c_0) \rangle, \langle s_5, \mathcal{B}_1, \mathcal{E}(c_1) \rangle, \dots)$ and $\sigma_2 = (\langle s_1, \mathcal{B}_0, \mathcal{E}(c_0) \rangle, \langle s_5, \mathcal{B}_1, \mathcal{E}(c_1) \rangle, \langle s_5, \{s_5\}, \rightarrow \circ \pi_0 \rangle, \langle s_8, \mathcal{B}_4, \pi_0 \rangle, \langle s_8, \{s_8\}, \varepsilon \rangle)$ are two runs of K from $\langle s_1, \mathcal{B}_0, \mathcal{E}(c_0) \rangle$. According to definition 6 and the assumption that $(\forall v \in \mathcal{V})\text{COST}(v) = 1$, $\text{AVOC}(\sigma_1) = \lim_{n \to \infty} 2n/(1 + 1 + 4n) = 0.5$ and $\text{AVOC}(\sigma_2) = 2/(1 + 2) \approx 0.67$. Furthermore, π_S is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$. For each $s \in \mathcal{I}$ and $\sigma' \in \text{RUNS}(\langle s, \mathcal{I}, \mathcal{E}(c_0) \rangle, K)$, $\text{AVOC}(\sigma') < 1 < |\mathcal{V}|$ (i.e., the cost of observation information acquisition required at every step of the execution of π_C).

4 Observation Reduction for Plans with Contexts

In this section, we will introduce an algorithm to observation reduction for general plans with contexts. The algorithm (i.e., OBSREDUCE) is reported in Figure 3. Given a planning domain $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{R} \rangle$, an observation setting $\langle \mathcal{V}, \mathcal{X} \rangle$ on Σ, and a plan with contexts $\pi_C = \langle \mathcal{C}', c'_0, act, ctxt \rangle$ for Σ, it transforms π_C into a structured plan $\pi_S = \langle \mathcal{C}, c_0, \mathcal{E} \rangle$, which is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and only branches on necessary observation information. We assume that $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ is a global variable.

```
1.  procedure OBSREDUCE(π_C)
2.    SC_A := SC_V := loops := ∅, C = ∅;
3.    dis := SIMULATE(I × {c'_0}, π_C);
4.    V_obs := REDUCE(dis, V);
5.    for all SC ∈ loops ∪ {I × {c'_0}}
6.      create a context c_SC;
7.      C := C ∪ {c_SC};
8.    for all c_SC ∈ C
9.      E[c_SC] := CONPLAN(SC, π_C, V_obs);
10.   return ⟨C, c_{I×{c'_0}}, E⟩;
11. end
```

Figure 3: OBSREDUCE algorithm

At every step (i.e., choosing an action) of the execution of π_C, the agent only needs to distinguish the current state–context pair $\langle s, c \rangle$ from $\langle s', c' \rangle \in \mathcal{SC}$ ($\mathcal{SC}$ is the set of possible current state–context pairs) such that:

- the action specified by π_C on $\langle s', c' \rangle$ is different from that on $\langle s, c \rangle$ (i.e., $act(s', c') \neq act(s, c)$); or
- the contexts corresponding to some state s'' will not be identical at next step (i.e., $\langle s, c \rangle \overrightarrow{a} \langle s'', c_1 \rangle$ and $\langle s', c' \rangle \overrightarrow{a} \langle s'', c_2 \rangle$ for some $a \in \mathcal{A}$ where $c_1 \neq c_2$).[3]

We can use two state–context pairs (e.g., $\langle \langle s, c \rangle, \langle s', c' \rangle \rangle$) to represent a basic distinguishing task. In the algorithm, all the basic distinguishing tasks that may arise during the execution of π_C from $\mathcal{I} \times \{c'_0\}$, are recorded in dis. On the other hand, π_C may result in iterative behaviors. Therefore, during the execution of π_C, some loops of sets of state–context pairs encountered may arise. In the algorithm, all the entrances of these loops, and all the sets of state–context pairs encountered, are recorded in $loops$ and $\mathcal{SC}_V$ respectively. $\mathcal{SC}_A$ records the sets of state–context pairs encountered in some runs of π_C (i.e., $\mathcal{SC}_A$ is the current set of state–context pairs). $loops$, $\mathcal{SC}_V$, and $\mathcal{SC}_A$ are all global variables.

Initially, the recursive procedure SIMULATE$(\mathcal{I} \times \{c'_0\}, \pi_C)$ is called to calculate dis (line 3). And then, REDUCE$(dis, \mathcal{V})$ is called to calculate $\mathcal{V}_{obs} \subseteq \mathcal{V}$ such that $(\forall \langle \langle s, c \rangle, \langle s', c' \rangle \rangle \in dis)(\exists v \in \mathcal{V}_{obs})\mathcal{X}(s, v) \neq \mathcal{X}(s', v)$ (line 4). So at every step of the execution of π_C from $\mathcal{I} \times \{c'_0\}$, the agent does not need to consider the values of the observation variables in $\mathcal{V} - \mathcal{V}_{obs}$. At last, for each $\mathcal{SC} \in loops \cup \{\mathcal{I} \times \{c'_0\}\}$, the algorithm creates a distinct context $c_{\mathcal{SC}}$ to represent the place where $\mathcal{E}[c_{\mathcal{SC}}]$ (i.e., the rest of the plan π_S to be executed form $\mathcal{SC}$) begins (lines 5–7), and calls the recursive procedure CONPLAN$(\mathcal{SC}, \pi_C, \mathcal{V}_{obs})$ to construct $\mathcal{E}[c_{\mathcal{SC}}]$ that only branches on some subset of $\mathcal{V}_{obs}$ (lines 8–9). The initial

[3] For example, consider the situation and plan π_C depicted in Example 1. Initially, it makes no sense to observe the value of any $v \in \mathcal{V}$ because $(\forall s \in \mathcal{I})(act(s, c'_0) = \rightarrow) \wedge (\forall s' \in \mathcal{R}(s, \rightarrow))ctxt(s, c'_0, s') = c'_0$. If we modify π_C such that $ctxt(s_0, c'_0, s_4) = c'_1$, then s_0 should be distinguished from s_1 because different actions (i.e., $\downarrow$ and $\leftarrow$) will be executed on s_4 at next step.

context that represents the place where the whole execution begins, is set as $c_{\mathcal{I}\times\{c'_0\}}$ (line 10).

SIMULATE, REDUCE, and CONPLAN subroutines are shown in Figure 4 and Figure 5. Now let us introduce them.

SIMULATE$(\mathcal{SC}, \pi_C)$ calculates the set of basic distinguishing tasks $dis \subseteq (\mathcal{S}\times\mathcal{C}')^2$ that may arise during the execution of π_C from $\mathcal{SC}$. Its basic idea is to simulate the behaviors induced by π_C from $\mathcal{SC}$. During the simulation, the entrances of loops (i.e., loops of sets of state–context pairs encountered) are recorded in *loops* (lines 13–14); and according to the actions specified by π_C and the successive state–context pairs, the possible current state–context pairs are dealt with separately (lines 18–24, and lines 28–38[4]).

Given $dis \subseteq (\mathcal{S}\times\mathcal{C}')^2$ and $\emptyset \subset \mathcal{V}' \subseteq \mathcal{V}$ as input, the procedure REDUCE returns $\mathcal{V}_{obs} \subseteq \mathcal{V}'$ such that $(\forall\langle\langle s,c\rangle,\langle s',c'\rangle\rangle \in dis)(\exists v \in \mathcal{V}_{obs})\mathcal{X}(s,v) \neq \mathcal{X}(s',v)$. It starts with $\mathcal{V}_{obs} = \emptyset$ and iteratively adds some $v \in \mathcal{V}'$ (whose observation cost per basic distinguishing task is minimal, see line 71) into $\mathcal{V}_{obs}$ until $d = dis$, where $d = \{\langle\langle s,c\rangle,\langle s',c'\rangle\rangle \in dis \mid (\exists v \in \mathcal{V}_{obs})\mathcal{X}(s,v) \neq \mathcal{X}(s',v)\}$.

For the execution of π_C started from $\mathcal{SC} \subseteq \mathcal{S}\times\mathcal{C}'$, CONPLAN$(\mathcal{SC}, \pi_C, \mathcal{V}_{obs})$ constructs the corresponding part of π_S, which only branches on some subset of $\mathcal{V}_{obs}$. VPLAN is similar to CONPLAN except that it checks whether $\mathcal{SC}$ is an entrance of a loop (line 61). In this case, the context corresponding to $\mathcal{SC}$ (i.e., $c_{\mathcal{SC}}$) is returned by VPLAN. The basic idea of CONPLAN is similar to that of SIMULATE, i.e., it is to simulate the behaviors induced by π_C from $\mathcal{SC}$. If two possible current state–context pairs in $\mathcal{SC}$ should be distinguished from each other, then the executions of π_C started from them, will be simulated separately (line 41, and lines 44–58).

We now show some properties of the algorithm OBSREDUCE(π_C). To do so, we first introduce the notion of SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$. SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$ is the minimal set satisfying the following rules:

- $\mathcal{I}\times\{c'_0\} \in$ SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$;
- if $\mathcal{SC} \in$ SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$, and DIVIDE$(\mathcal{SC}, \pi_C) = \{\langle a, \mathcal{SC}\rangle\}$ for some $a \in \mathcal{A}$, then EXECUTE$(\mathcal{SC}, a, \pi_C) \in$ SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$;
- if $\mathcal{SC} \in$ SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$, then $(\forall\langle a, \mathcal{SC}'\rangle \in$ DIVIDE$(\mathcal{SC}, \pi_C))\mathcal{SC}' \in$ SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$.

In the execution of OBSREDUCE(π_C), SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)$ is recorded as $\mathcal{SC}_V$, which is updated by the calls of SIMULATE (line 17, Figure 4).

Theorem 1 *Let $K = \langle\mathcal{SC}, \mathcal{SC}_0, \mathcal{TR}\rangle$ be the execution structure of a plan with contexts π_C on Σ.* OBSREDUCE(π_C) *is guaranteed to terminate, and it is polynomial in $|\mathcal{V}|$, $|$SCs$(\mathcal{I}\times\{c'_0\}, \pi_C)|$, and $|\mathcal{SC}|$.*

Proof: See appendix. ■

Theorem 2 *When* OBSREDUCE(π_C) *returns a structured plan $\pi_S = \langle\mathcal{C}, c_0, \mathcal{E}\rangle$ (let K be the execution struc-*

[4]Formally, let $\langle a, \mathcal{SC}'\rangle \in$ DIVIDE$(\mathcal{SC}, \pi_C)$ and $\langle s,c\rangle \in \mathcal{SC}'$ then $(\forall\langle s',c'\rangle \in \mathcal{SC})(\langle s',c'\rangle \in \mathcal{SC}') \leftrightarrow (act(s',c') = a) \wedge (\forall s'' \in \mathcal{R}(s,a) \cap \mathcal{R}(s',a))(ctxt(s,c,s'') = ctxt(s',c',s''))$. VAILD$(\mathcal{SC}, \pi_C) = \{\langle s,c\rangle \in \mathcal{SC} | act(s,c)$ is defined in $\pi_C\}$. And EXECUTE$(\mathcal{SC}, a, \pi_C) = \{\langle s,c\rangle | (\exists\langle s',c'\rangle \in \mathcal{SC})\langle s',c'\rangle \overrightarrow{a} \langle s,c\rangle\}$.

```
12.  procedure SIMULATE(SC, π_C)
13.    if SC ∈ SC_A then
14.      loops := loops ∪ {SC}, return ∅;
15.    if SC ∈ SC_V or VAILD(SC, π_C) = ∅
16.      then return ∅;
17.    SC_A := SC_A ∪ {SC}, SC_V := SC_V ∪ {SC};
18.    divitions := DIVIDE(SC, π_C);
19.    if divitions = {⟨a, SC⟩} for some a ∈ A then
20.      return SIMULATE(EXECUTE(SC, a, π_C), π_C);
21.    dis = ∅, SC' := SC;
22.    for all ⟨a', SC''⟩ ∈ divitions
23.      SC' := SC' − SC'';
24.      dis := dis ∪ (SC'' × SC') ∪ SIMULATE(SC'', π_C)
25.    SC_A := SC_A − {SC};
26.    return dis;
27.  end

28.  procedure DIVIDE(SC, π_C)
29.    divisions := ∅, SC' := VALID(SC, π_C);
30.    while SC' ≠ ∅ do
31.      select a pair ⟨s, c⟩ from SC' randomly;
32.      divition := {⟨s, c⟩}, SC' := SC' − {⟨s, c⟩};
33.      for all ⟨s', c'⟩ ∈ SC' such that act(s', c') = act(s, c)
         and ∀s'' ∈ R(s, act(s, c)) ∩ R(s', act(s', c')).
         ctxt(s, c, s'') = ctxt(s', c', s'')
34.        division := division ∪ {⟨s', c'⟩};
35.        SC' := SC' − {⟨s', c'⟩};
36.      divisions := divisions ∪ {⟨act(s, c), divition⟩};
37.    return divisions;
38.  end

39.  procedure CONPLAN(SC, π_C, V_obs)
40.    if VALID(SC, π_C) = ∅ then return ε;
41.    divitions := DIVIDE(SC, π_C);
42.    if divitions = {⟨a, SC⟩} for some a ∈ A then
43.      return a∘VPLAN(EXECUTE(SC, a, π_C), π_C, V_obs);
44.    dis = ∅, SC' := SC;
45.    for all ⟨a', SC''⟩ ∈ divitions
46.      SC' := SC' − SC'', dis := dis ∪ (SC'' × SC');
47.    V_now := REDUCE(dis, V_obs);
48.    for 1 ≤ i ≤ |divitions|
49.      select a pair ⟨a', SC''⟩ from divisions randomly;
50.      divisions := divisions − {⟨a', SC''⟩};
51.      o_i := OBS({s|∃c.⟨s, c⟩ ∈ SC''}, V_now, X);
52.      π_i :=VPLAN(SC'', π_C, V_obs);
53.    n := |divisions|, SC'' := SC − VALID(SC, π_C);
54.    if SC'' ≠ ∅ then
55.      n := n + 1;
56.      o_n := OBS({s|∃c.⟨s, c⟩ ∈ SC''}, V_now, X);
57.      π_n := ε;
58.    return switch(V_now){⟨o_i, π_i⟩|1 ≤ i ≤ n};
59.  end

60.  procedure VPLAN(SC, π_C, V_obs)
61.    if SC ∈ loops then return c_SC;
62.    return CONPLAN(SC, π_C, V_obs);
63.  end
```

Figure 4: Subroutines: SIMULATE,DIVIDE and CONPLAN

```
64.  procedure REDUCE(dis, V')
65.    for all v ∈ V' do di[v] := ∅;
66.    for all ⟨⟨s, c⟩, ⟨s', c'⟩⟩ ∈ dis, and v ∈ V'
67.      if X(s, v) ≠ X(s', v) then
68.        di[v] := di[v] ∪ {⟨⟨s, c⟩, ⟨s', c'⟩⟩};
69.    V_obs := ∅;
70.    while dis ≠ ∅
71.      find v ∈ V' such that COST(v)/|di[v]| =
           MIN({COST(v')/|di[v']| | v' ∈ V'});
72.      V_obs := V_obs ∪ {v};
73.      for all v' ∈ V
74.        di[v'] := di[v'] − di[v];
75.      dis := dis − di[v];
76.    return V_obs;
77.  end
```

Figure 5: REDUCE subroutine

ture corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and π_S), π_S is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$, and $(\forall s \in \mathcal{I})(\forall \sigma \in \text{RUNS}(\langle s, \mathcal{I}, \mathcal{E}(c_0) \rangle, K))\text{AVOC}(\sigma) \leq \sum_{v \in \mathcal{V}} \text{COST}(v)$.

Proof: See appendix. ■

Theorem 2 says that the behaviors induced by π_S are similar to those induced by π_C, and the observation cost required at every step of the execution of π_S, is not more than that of π_C (i.e., $\sum_{v \in \mathcal{V}} \text{COST}(v)$). In the following example, we illustrate the algorithms described in this section.

Example 3 *Consider the situation $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$ and the plan with contexts π_C depicted in Example 1. We apply the algorithm* OBSREDUCE *to π_C. Firstly, dis is computed by* SIMULATE$(\mathcal{I} \times \{c'_0\}, \pi_C)$*, and it is $\{\langle\langle s_3, c'_0\rangle, \langle s_5, c'_0\rangle\rangle, \langle\langle s_4, c'_0\rangle, \langle s_5, c'_0\rangle\rangle, \langle\langle s_5, c'_1\rangle, \langle s_8, c'_0\rangle\rangle\}$ (at the same time, $loops = \{\{\langle s_3, c'_0\rangle, \langle s_4, c'_0\rangle, \langle s_5, c'_0\rangle\}\}$); and then, $\mathcal{V}_{obs}$ is computed by* REDUCE$(dis, \mathcal{V})$*, and it is $\{E,S\}$; lastly, the structured plan π_S shown in Example 2 is constructed and returned, where $c_0 = c_{\mathcal{I} \times c'_0}$ and $c_1 = c_{\{\langle s_3, c'_0\rangle, \langle s_4, c'_0\rangle, \langle s_5, c'_0\rangle\}}$.*

5 Conclusions and Future Work

In this paper we present a first attempt to solve the problem of observation reduction for general plans with contexts. At first, we define structured plans that can encode sequential, conditional and iterative behaviors, and are expressive enough for dealing with incomplete observation information and with internal states of the agent. Intuitively, structured plans are similar to programs with conditional statements and loops, in which contexts are used to represent the destination points of jumps occurring in execution control. We also define what does "a structured plan is equivalent to a plan with contexts" mean, and the expected cost of observation information acquisition required at every step of the execution of a structured plan. And then, we give an algorithm (i.e., OBSREDUCE) to the problem. Given a planning domain $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{R} \rangle$, an observation setting $\langle \mathcal{V}, \mathcal{X} \rangle$ on Σ, and a plan with contexts $\pi_C = \langle \mathcal{C}', c'_0, act, ctxt \rangle$ for Σ, then:

1. OBSREDUCE(π_C) terminates, and is polynomial in $|\mathcal{V}|$, $|\text{SCS}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ (i.e., the number of sets of state–context pairs possibly encountered), and $|\mathcal{SC}|$ (i.e., the number of state–context pairs possibly encountered);
2. when OBSREDUCE(π_C) returns a structured plan π_S, π_S is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X} \rangle$, and the expected cost of observation information acquisition required at every step of the execution of π_S is not more than $\sum_{v \in \mathcal{V}} \text{COST}(v)$ (i.e., the cost of observation information acquisition required at every step of the execution of π_C).

In this paper, we assume a setting of strict uncertainty in which the space of possible effects of actions is known, but the probabilities of these potential alternatives can not be quantified. An extension of our work to the settings where a probability distribution over the set of possible effects of actions is available (e.g., see [Kaelbling *et al.*, 1998; Bonet and Geffner, 2000]), is one of the main objectives of our future research.

The definition of structured plans, can work as a basis of a new framework (for planning with extended goals under partial observability) with respect to the one proposed in [Bertoli *et al.*, 2003; Bertoli and Pistore, 2004], because $\langle \mathcal{V}, \mathcal{X} \rangle$ can be used to model partial observability. So we plan to define a procedure (which adapts to the idea of observation and context reduction) for planning with extended goals and partial observability in the new framework.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant No.60773201, Guangdong Natural Science Foundation under grant No.07006474, and Guangdong Scientific and Technological Project Foundation under grant No.2007B01020044.

References

[Bertoli and Pistore, 2004] Piergiorgio Bertoli and Marco Pistore. Planning with extended goals and partial observability. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pages 270–278, 2004.

[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, 2001.

[Bertoli *et al.*, 2003] Piergiorgio Bertoli, Alessandro Cimatti, Marco Pistore, and Paolo Traverso. A framework for planning with extended goals under partial observability. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, pages 215–225, 2003.

[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, 2000.

[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.

[Huang *et al.*, 2007] Wei Huang, Zhonghua Wen, Yunfei Jiang, and Lihua Wu. Observation reduction for strong plans. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1930–1935, 2007.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence Research*, 101(1–2):99–134, 1998.

[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, 2002.

[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, 2001.

[Sardina *et al.*, 2006] Sebastian Sardina, Giuseppe De Giacomo, Yves Lesperance, and Hector J. Levesque. On the limits of planning over belief states under strict uncertainty. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pages 463–471, 2006.

A Proofs

Theorem 1 *Let $K = \langle \mathcal{SC}, \mathcal{SC}_0, \mathcal{TR} \rangle$ be the execution structure of a plan with contexts π_C on Σ.* ObsReduce(π_C) *is guaranteed to terminate, and it is polynomial in $|\mathcal{V}|$, $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$, and $|\mathcal{SC}|$.*

Proof: Firstly, we want to prove that the numbers of the calls of Simulate, ConPlan, and VPlan occurring in the execution of ObsReduce(π_C), are finite and polynomial in $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$:

- $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ is finite because it is a subset of $2^{\mathcal{S} \times \mathcal{C}}$.
- According to Figure 4, only the sets of state–context pairs in $\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)$ are passed to Simulate, ConPlan, and VPlan. On the other hand, for each $\mathcal{SC}' \in \text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)$, there are at most $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ sets of state–context pairs that can reach to it directly. Therefore, there are at most $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ calls of Simulate, ConPlan, and VPlan that are on $\mathcal{SC}'$ (see lines 13–17, line 25, and line 61).

Secondly, we will prove that, in every call of Simulate, Divide, ConPlan, VPlan, and Reduce, all the computation processes (besides the new calls of these procedures) terminate, and are polynomial in $|\mathcal{V}|$, $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ and $|\mathcal{SC}|$:

1. According to Figure 4 (line 18, and 41), only the sets of state–context pairs in $\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)$ are possibly passed to Divide. It is easy to find that each call of Divide terminates and is polynomial in $|\mathcal{SC}|$.
2. $(\forall \mathcal{SC}' \in \text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C))(\forall \langle s, c\rangle, \langle s', c'\rangle \in \mathcal{SC}')(s = s') \rightarrow (c = c')$ (see Figure 4, line 33), because $(\forall \langle s, c\rangle, \langle s', c'\rangle \in \mathcal{I} \times \{c'_0\})(s = s') \rightarrow (c = c')$.
3. For each $\langle dis, \mathcal{V}'\rangle$ passed to some call of Reduce, we have $(\forall \langle \langle s, c\rangle, \langle s', c'\rangle\rangle \in dis)(\exists \mathcal{SC}' \in \text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C))\{\langle s, c\rangle, \langle s', c'\rangle\} \subseteq \mathcal{SC}'$ (see line 4, and line 47). So $(\forall \langle \langle s, c\rangle, \langle s', c'\rangle\rangle \in dis)s \neq s'$ (according to item 2 shown above). Consequently, under the assumption that any two distinct states can be distinguished by one observation variable at least, each call of Reduce terminates and is polynomial in $|\mathcal{V}|$ and $|\mathcal{SC}|$.
4. It is easy to find that, in every call of Simulate, ConPlan, and VPlan, all the computation processes (besides the recursive calls of themselves) terminate, and are polynomial in $|\mathcal{V}|$, $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ and $|\mathcal{SC}|$.

Lastly, there are $|loops \cup \{\mathcal{I} \times \{c'_0\}\}| \leq |\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$ iterations of the loops shown in lines 5–9, Figure 3. Therefore ObsReduce(π_C) is guaranteed to terminate, and is polynomial in $|\mathcal{V}|$, $|\text{SCs}(\mathcal{I} \times \{c'_0\}, \pi_C)|$, and $|\mathcal{SC}|$. ■

Theorem 2 *When* ObsReduce(π_C) *returns a structured plan $\pi_S = \langle \mathcal{C}, c_0, \mathcal{E}\rangle$ (let K be the execution structure corresponding to $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$ and π_S), π_S is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$, and $(\forall s \in \mathcal{I})(\forall \sigma \in \text{Runs}(\langle s, \mathcal{I}, \mathcal{E}(c_0)\rangle, K))\text{AvOC}(\sigma) \leq \sum_{v \in \mathcal{V}} \text{Cost}(v)$.*

Proof: π_S is constructed by simulating the behaviors induced by π_C from $\mathcal{I} \times \{c'_0\}$ (see figure 3, figure 4, and figure 5). At each step of the simulation process, only one of the four cases shown below is possible (suppose $\langle s, c\rangle$ is the current state–context pair, $\mathcal{SC}'$ ($\langle s, c\rangle \in \mathcal{SC}'$) is the set of possible current state–context pairs):

1. Valid($\mathcal{SC}', \pi_C$) $= \emptyset$, no observation is made, and the execution terminates (line 40, figure 4);
2. $(\exists a \in \mathcal{A})(\forall \langle s', c'\rangle \in \mathcal{SC}')act(s', c') = a$, no observation is made, and a is selected to execute (lines 42–43, figure 4);
3. $\langle s, c\rangle \notin \text{Valid}(\mathcal{SC}', \pi_C) \neq \emptyset$, the values of the observation variables in $\mathcal{V}' \subseteq \mathcal{V}$ are observed where $(\forall s' \in \{s''|(\exists c'')\langle s'', c''\rangle \in \text{Valid}(\mathcal{SC}', \pi_C)\})(\exists v \in \mathcal{V}')\mathcal{X}(s, v) \neq \mathcal{X}(s', v)$, and the execution terminates (lines 47, 53–57, figure 4);
4. $\langle s, c\rangle \in \text{Valid}(\mathcal{SC}', \pi_C)$, $\mathcal{SC}' - \{\langle s', c'\rangle \in \mathcal{SC}'|act(s', c') = act(s, c)\} \neq \emptyset$, the values of the observation variables in $\mathcal{V}' \subseteq \mathcal{V}$ are observed where $(\forall s' \in \{s''|(\exists c'')(\langle s'', c''\rangle \in \mathcal{SC}') \wedge \neg(act(s'', c'') = act(s, c))\})(\exists v \in \mathcal{V}')\mathcal{X}(s, v) \neq \mathcal{X}(s', v)$, and $act(s, c)$ is selected to execute (lines 47–52, figure 4).

It is easy to find that π_S is equivalent to π_C on $\langle \Sigma, \mathcal{V}, \mathcal{X}\rangle$.

According to the four cases shown before, we can also find that for each $v \in \mathcal{V}$ and each step of the execution of π_S, v can not be evaluated more than once (see procedure Divide in figure 4). So for all $s \in \mathcal{I}$ and $\sigma \in \text{Runs}(\langle s, \mathcal{I}, \mathcal{E}(c_0)\rangle, K)$, $\text{AvOC}(\sigma) \leq \sum_{v \in \mathcal{V}} \text{Cost}(v)$. ■