Supplementary Material: Finding plans subject to stipulations on what information they divulge

Yulin Zhang\textsuperscript{1}, Dylan A. Shell\textsuperscript{1} and Jason M. O’Kane\textsuperscript{2}

\textsuperscript{1} Texas A&M University, College Station TX, USA
\textsuperscript{2} University of South Carolina, Columbia SC, USA

A Generalized notation for label maps and their preimages

The notation associated with a label map \( h \), and its preimage \( h^{-1} \), is extended to the set of events, execution, set of executions, and p-graphs in the natural way. In detail, for \( h : Y \cup U \to X \), we introduce the following:

**Events** Given any set of events \( L \subseteq Y \cup U \), its image is \( h[L] = \{ h(\ell) \mid \ell \in L \} \). Conversely, for set \( L' \subseteq X \), its preimage is \( h^{-1}[L'] = \{ \ell \in Y \cup U \mid h(\ell) \in L' \} \).

**Executions** Given any execution \( s = \ell_0\ell_1\ldots\ell_k \), where \( \ell_i \in Y \cup U \), its image is \( h(s) = h(\ell_0)h(\ell_1)\ldots h(\ell_k) \), and for any execution \( s' = \ell'_0\ell'_1\ldots\ell'_k \), where \( \ell'_i \in X \), its preimage is \( h^{-1}(s') = \{ s \mid h(s) = s' \} \).

**Sets of executions** Given any set of executions \( A \), where \( \forall s \in A, s \in (Y \cup U)^* \), its image is \( h[A] = \{ h(s) \mid s \in A \} \). Conversely for any set of executions \( A' \), where \( \forall s' \in A', s' \in X^* \), its preimage is \( h^{-1}[A'] = \{ s \mid h(s) \in A' \} \).

**P-graphs** Given any p-graph \( G = (V_u \cup Y_u, Y, U, V_0) \), its image p-graph \( h(G) = (V_u \cup Y_u, h[Y], h[U], V_0) \) is produced by replacing the set of events \( L \) on each edge \( e \) with \( h[L] \). Analogously, given p-graph \( G = (V_u \cup Y_u, X_y, X_u, V_0) \), its preimage p-graph \( h^{-1}(G) = (V_u \cup Y_u, h^{-1}[X_y], h^{-1}[X_u], V_0) \) is constructed by replacing the set of events \( L' \) on each edge \( e \) with \( h^{-1}[L'] \).

B Characteristics of I-state graphs and label maps

B.1 The effects of I-state graph and label map structure on the associated sets

In this section, we present proofs of the first set of properties used in the paper. They deal with how both the I-state and label maps affect the information correspondence between the observer and the world. Some ancillary lemmas are needed to establish the results, these have been given a distinct numbering scheme to set them apart as subordinate.

**Lemma B-3.** For any event \( \ell \in Y \cup U \), \( h^{-1} \circ h(\ell) \supseteq \{ \ell \} \). Similarly, we have \( \forall L \subseteq Y \cup U \), \( h^{-1} \circ h[L] \supseteq L \) and \( \forall s = \ell_0\ell_1\ldots\ell_n \in (Y \cup U)^* \), \( s \in h^{-1} \circ h(s) \).

**Proof.** First, we are going to prove \( \forall \ell \in Y \cup U \), \( h^{-1} \circ h(\ell) \supseteq \{ \ell \} \). Since \( h \) is a function, there are two cases for the images of the events in \( G \): First, \( \forall \ell_1, \ell_2 \in U \cup Y \), \( h(\ell_1) \neq h(\ell_2) \). In this case, no two events are mapped to the same output. In other words, each image element has a unique preimage, \( \{ \ell \} = h^{-1}(h(\ell)) \). Secondly, \( \exists \ell_1, \ell_2 \in U(G) \cup Y(G) \), \( h(\ell_1) = h(\ell_2) \). Then we have \( h^{-1}(h(\ell_1)) = h^{-1}(h(\ell_2)) = \ell_1 \cup \ell_2 \), \( \{ \ell_1 \} \subset h^{-1}(h(\ell_1)) \), and \( \{ \ell_2 \} \subset h^{-1}(h(\ell_2)) \). Hence, \( h^{-1} \circ h(\ell) \supseteq \{ \ell \} \).
Next, following the result of $h^{-1} \circ h(\ell) \supseteq \{\ell\}$, we have that $h^{-1} \circ h[L] = \cup_{\ell \in L} h^{-1} \circ h[\ell] \supseteq L$ for any $L \subseteq Y \cup U$.

Finally, we will prove $s \in h^{-1} \circ h(s)$ by induction for all $s = \ell_0 \ell_1 \ldots \ell_k \in (Y \cup U)^*$. Let $s^k = \ell_0 \ell_1 \ldots \ell_k$ be the prefix of $s$ with length $k + 1$, where $0 \leq k < n$. When $k = 0$, $s^0$ only contains an action or observation and, we have $s^0 \in h^{-1} \circ h(s^0)$. Suppose $s^k = \ell_0 \ell_1 \ldots \ell_k \in h^{-1} \circ h(s^k)$ holds for $k$. The inductive step: $h^{-1} \circ h(s^{k+1}) = \cup_{\ell_0 \ell_1 \ldots \ell_k \in h^{-1} \circ h(s^k)} \ell_0 \ell_1 \ldots \ell_k \ell_{k+1}$, which is since $\ell_0 \ell_1 \ldots \ell_k \in h^{-1} \circ h(s^k)$ and $\ell_{k+1} \in h^{-1} \circ h(s^{k+1})$. Hence, $s_{k+1} \in h^{-1} \circ h(s^{k+1})$. Therefore, $s \in h^{-1} \circ h(s), \forall s \in (Y \cup U)^*$.

**Lemma B-4.** For any $\ell' \in X$, $h \circ h^{-1}(\ell') = \{\ell'\}$. Similarly, we have $\forall L' \subseteq X$, $h \circ h^{-1}[L'] = L'$ and $\forall s = \ell_0' \ell_1' \ldots \ell_k' \in X^*$, $h \circ h^{-1}(s) = \{s\}$.

**Proof.** Firstly, we will prove $h \circ h^{-1}(\ell') = \{\ell'\}$ holds for any $\ell' \in X$. Let $h^{-1}(\ell') = \{l \in Y \cup U | h(l) = \ell'\}$. Then $\forall \ell \in h^{-1}(\ell')$, we have $h(\ell) = \ell'$. Therefore, $h \circ h^{-1}(\ell') = \{\ell'\}$.

Following from $h \circ h^{-1}(\ell') = \{\ell'\}$, we have $h \circ h^{-1}[L'] = L'$ for any $L' \subseteq X$.

Thirdly, we will prove $h \circ h^{-1}(s) = \{s\}$ by induction for any $s = \ell_0' \ell_1' \ldots \ell_k' \in X^*$. Let $s^k$ be the prefix of $s$ with length $k + 1$, where $0 \leq k < n$. When $k = 0$, $s^0$ only contains an action or observation and, we have $\{s^0\} = h \circ h^{-1}(s^0)$. Suppose $\{s^k\} = h \circ h^{-1}(s^k)$ holds for $k$. Then $h \circ h^{-1}(s^{k+1}) = \cup_{\ell_0' \ell_1' \ldots \ell_k' \in h^{-1}(s^k)} \ell_0' \ell_1' \ldots \ell_{k+1}'$, which is since $\ell_0' \ell_1' \ldots \ell_k' \in h^{-1}(s^k)$ and $\ell_{k+1}' \in h^{-1}(s^{k+1})$. Hence, $h \circ h^{-1}(s^{k+1}) = \{s^{k+1}\}$. Therefore, $\forall s \in X^*$, $h \circ h^{-1}(s) = \{s\}$.

The preceding two lemmas show that, because $h$ is a function, it can either preserve information (when it is a bijection) or it can lose information (by mapping multiple inputs to a single output). The loss of information is felt in $Y \cup U$ by the extent to which some $\{z\} \in Y \cup U$ grows under $h^{-1} \circ h$. In contrast, starting from $x \in X$, the uncertainty, measurable via set cardinality under $h^{-1}$, is washed out again when pushed forward to $X$.

These two general lemmas set the stage for more specific properties of I-state graphs.

**Lemma 1 (Properties of I).** Given $I$ and $h$, the following hold:

i. $I = h \circ h^{-1}(I)$.

ii. $\forall s' \in L(I), \forall s \in h^{-1}(s'), V^I_s = V^I_{h^{-1}(s)}$.

iii. $L(h^{-1}(I)) = h^{-1}[L(I)]$.

iv. $\forall B \subseteq V(I), h^{-1}([B]) = s^B_{h^{-1}(I)}$.

**Proof (Property i.).** According to Lemma B-4, each event set $L$ in $I$ will not change when we apply operation $h \circ h^{-1}$ on $I$. Therefore, we have $I = h \circ h^{-1}(I)$ by replacing every set of events $L$ in $I$ with $h \circ h^{-1}(L)$.

**Proof (Property ii.).** We need to prove that $s'$ and its preimage $s$ reach the same set of vertices in $I$ and $h^{-1}(I)$ respectively. According to the construction of $h^{-1}(I)$, we have $\forall s' \in L(I), \forall s \in h^{-1}(s'), V^I_s = V^I_{h^{-1}(s')}$. Next, we will prove $\forall s' \in L(I), \forall s \in h^{-1}(s'), V^I_s \supseteq V^I_{h^{-1}(s')}$ by contradiction. Suppose $\exists s' \in L(I), \exists s \in h^{-1}(s'), V^I_s \not\supseteq V^I_{h^{-1}(s')}$.
Fig. B-8: Both the label map and the I-state graph can degrade information. Leftmost: a scenario where the world p-graph $W$, a plan $P$, and divulged plan information $D$ are all identical. Second from the left: an I-state graph $I$ with the same structure as $W$ and an identity label map $h$. Second from the right: an I-state graph with $I$ the same structure as $W$ and a label map $h$ which conveys some actions/observations. Rightmost: both $h$ and $I$ degrade information independently.

$V_s^{h^{-1}(I)}$. Then we have $V_s' \subseteq V_s^{h^{-1}(I)}$. If $s$ is the preimage of only $s'$, then we should have, according to the construction of $h^{-1}(I)$, $V_s' = V_s^{h^{-1}(I)}$ instead. Hence, $s$ is the preimage of at least two different executions $s'$ and $s''$, which contradicts with the fact that $h$ is a function. Therefore, $\forall s' \in \mathcal{L}(I), \forall s \in h^{-1}(s'), V_s' = V_s^{h^{-1}(I)}$.

$\forall s' \in \mathcal{L}(I), \forall s \in h^{-1}(s'), s'$ reach the same set of vertices as those reached by $s$ in $h^{-1}(I)$. Since each vertex in $V_s'$ is isomorphic to the same one in $V_s^{h^{-1}(I)}$, we have $V_s'$ is identical to $V_s^{h^{-1}(I)}$.

**Proof (Property iii.).** $\implies$: Given any execution $s$ from p-graph $h^{-1}(I)$, we will prove $h(s)$ is an execution from p-graph $I$. According to Lemma 1.1, $I = h(h^{-1}(I))$. Thus, $h(s)$ is an execution on $I = h(h^{-1}(I))$. And we have $\mathcal{L}(h^{-1}(I)) \subseteq h^{-1}[\mathcal{L}(I)]$.

$\iff$: Given any execution $s \in h^{-1}[\mathcal{L}(I)]$, we will prove $s \in \mathcal{L}(h^{-1}(I))$. For any $s \in h^{-1}[\mathcal{L}(I)]$, we have $h(s)$ is an execution from p-graph $I$. According to Lemma 1.1, the set of vertices reached by $h(s)$ in p-graph $I$ is isomorphic to the set of vertices reached by $s'$ in $h^{-1}(h(s))$ in $h^{-1}(I)$. Hence, $s$ is an execution in $h^{-1}(I)$. Therefore, $\mathcal{L}(h^{-1}(I)) \supseteq h^{-1}[\mathcal{L}(I)]$.

**Proof (Property iv.).** $\implies$: Given any execution $s \in h^{-1}[\mathcal{S}_B^I]$, then we have $h(s) \in \mathcal{S}_B^I$ and $V_{s(h(s))}^I = B$. According to Lemma 1.2, we have $V_s^{h^{-1}(I)} = V_{h(s)}^I = B$. Hence, $s \in \mathcal{S}_B^{h^{-1}(I)}$.

$\iff$: Given any execution $s \in \mathcal{S}_B^{h^{-1}(I)}$, then we have $V_s^{h^{-1}(I)} = B$. According to Lemma 1.2, $V_{s(h(s))}^I = V_s^{h^{-1}(I)} = B$. Hence, $h(s) \in \mathcal{S}_B^I$ and therefore, we now have $s \in h^{-1}[\mathcal{S}_B^I]$.

$^5$ Since, by $h^{-1}(I)$ we refer to the graph $I$ with each of the edge labels replaced by preimages under $h$, there is a one-to-one correspondence between the two graphs via a natural isomorphism. For convenience we speak of the “same” vertex rather than being explicit about the associated bijection and, further, we have used ‘$\equiv$’ rather than ‘$\simeq$’.
B.2 How both label maps and I-state graphs erode information

Figure B-8 provides a visual example that shows how information can be degraded by a label map $h$, an I-state graph $I$, and both together. The first gives a scenario by providing a world p-graph $W$, a plan $P$, and divulged plan information $D$—all three are identical. The second figure shows an I-state graph with the same structure as $W$ and an identity label map. Every I-state corresponds to a single world state in this case. In the third figure, there is an I-state graph with the same structure as $W$, thus clearly possessing sufficient structure to account for the world states. But here a label map conflates some actions and some observations. A consequence is that the world states $w_1$ and $w_2$ are indistinguishable given I-state $i_1$ and plan $P$. In the last figure both $h$ and $I$ degrade information and do so independently. In this case, $w_3$ and $w_4$ are indistinguishable owing to the label map, $w_5$ and $w_6$ are indistinguishable owing to the collapsed structure in $I$.

C Plan information divulged in the image space is weaker

In this paper we have assumed that the information divulged to the observer about the robot’s execution is in the preimage space. This modeling decision may seem strange at first blush so this section provides some explanation and justification for it. As the observer will only see things in the image space, it may seem that granting access to information in the preimage or the image space would have little difference. But, since inference occurs by pulling back observed events to preimage space and then taking an intersection, there can be an appreciable difference. As this paper is interested in a worst-case adversarial conditions, we are interested in what strong ne’er-do-well observers might infer and thus study the problem where the adversary gains the maximum possible. From this setting the other variant can be easily posed as well (one simply needs to consider $h^{-1} \circ h(D)$ to simulate the knowledge of image space information).

To clarify the previous statements, we formalize the image space inference process:

**Definition C-16.** Given an I-state graph $I$, divulged information about the robot’s behavior $h(D)$ in the image space, and label map $h$, the set of estimated world states for I-states $B \subseteq V(I)$ is $W_B^{I,h(D)} = \{w \in V(W)|h^{-1}[S_B^I] \cap \mathcal{L}(h^{-1} \circ h(D)) \cap S_w^W \neq \emptyset\}$.

**Lemma C-5.** Given any p-graph $D$, $\mathcal{L}(D) \subseteq \mathcal{L}(h^{-1} \circ h(D))$.

**Proof.** According to Lemma B-3, for event $\ell \in \mathcal{L}(D)$, we have $\{\ell\} \subseteq \mathcal{L}(h^{-1} \circ h(\ell))$. Then the set of events bearing in each edge of p-graph $h^{-1} \circ h(D)$ is a superset of the corresponding edge in p-graph $D$. Therefore, $\mathcal{L}(D) \subseteq \mathcal{L}(h^{-1} \circ h(D))$. \hfill $\Box$

**Theorem C-3.** Given I-state graph $I$, divulged information $D$, world graph $W$, and label map $h$, the set of estimated world states for any set of I-states $B \subseteq V(I)$ is $W_B^{I,h(D)}$. By replacing $D$ with its image graph $h(D)$, the set of estimated world states for $B$ is $W_B^{I,h(D)}$. $\forall B \subseteq V(I), W_B^{I,h(D)} \supseteq W_B^{I,D}$.

**Proof.** According to Lemma C-5, $\mathcal{L}(D) \subseteq \mathcal{L}(h^{-1} \circ h(D))$. Thus $\forall B \subseteq V(I), \forall w \subseteq V(W)$, we have $h^{-1}[S_B^I] \cap \mathcal{L}(D) \cap S_w^W \subseteq h^{-1}[S_B^I] \cap \mathcal{L}(h^{-1} \circ h(D)) \cap S_w^W$. If $h^{-1}[S_B^I] \cap \mathcal{L}(D) \cap S_w^W \neq \emptyset$, then $h^{-1}[S_B^I] \cap \mathcal{L}(h^{-1} \circ h(D)) \cap S_w^W \neq \emptyset$. Thus, if $v \in W_B^{I,D}$, then $v \in W_B^{I,h(D)}$. Hence, $\forall B \subseteq V(I)$, we have $W_B^{I,h(D)} \supseteq W_B^{I,D}$. \hfill $\Box$
Fig. C-9: Find the estimated world states when given world graph \( W \), I-state graph \( I \), divulged graph \( D \) or \( h(D) \), label map \( h = \{ a_1, a_2 \mapsto \alpha; o_1, o_2 \mapsto o \} \).

Formula \( \rightarrow \) Clause_1 \& \ldots \& Clause_n
Clause \( \rightarrow \) Literal_1 \lor \ldots \lor Literal_m
Literal \( \rightarrow \) Symbol | \neg Symbol
Symbol \( \rightarrow v_0, v_1, v_2, \ldots \)

\[
\text{[VALUE]} \quad \langle \nu_i \rangle \Downarrow \text{eval}(\nu_i \in W_{I,D}^D)
\]
\[
\text{[NOT]} \quad \langle \nu_i \rangle \Downarrow \neg w
\]
\[
\langle \ell_1 \lor \ell_2 \rangle \Downarrow \text{the logical or of } w_1 \text{ and } w_2
\]
\[
\langle \ell_1 \land \ell_2 \rangle \Downarrow \text{the logical and of } w_1 \text{ and } w_2
\]

Fig. D-10: The syntax and natural semantics of the information stipulations, where \( \ell_i \), \( \nu_i \), represent a clause, literal, and symbol, respectively, and \( w_i \) is the result of the evaluation. The transition \( \langle e \rangle \Downarrow w \) denotes a transition, where \( e \) is any expression defined by the grammar and \( w \) is the value yielded by the expression.

An example to illustrate Theorem C-3 is shown in Figure C-9. Given \( W \) and \( I \),
\( W^I_{\{i_0\}} = \{ w_1 \} \), while \( W^I_{\{i_0\}, h(D)} = \{ w_1, w_2 \} \). Hence, \( W^I_{\{i_0\}, h(D)} \supseteq W^I_{\{i_0\}} \).

**D Stipulating properties on disclosed information**

The stipulation constraining the information to be disclosed is written as a propositional formula \( \Phi \). We give the syntax and semantics of these formulas briefly.

The formula \( \Phi \) is written in conjunctive normal form, consisting of symbols, literals and clauses as shown in Fig. D-10. Firstly, a basic, atomic symbol \( \nu_i \) is associated with each world state \( \nu_i \in V(W) \). If \( \nu_i \) is contained in the observer’s estimates \( W_{I,D}^I \), we will evaluate the corresponding symbol \( \nu_i \) as True. Otherwise, it evaluates as False. With each symbol grounded, we can evaluate literals and clauses compositionally, using logic operators \( \text{NOT, AND, OR} \). These are defined naturally, eventually enabling evaluation of \( \Phi \) on the observer’s estimate \( W^I_{\{i_0\}} \).

One version of our implementation, as is reported in Section 7, made use of a CTL model checker (nuXmv). In those cases, the tool was able to evaluate the formulas for us, via the sets \( W^I_{\{i_0\}} \) for grounding. In other implementations (also reported) we used our own evaluator.
We note that the semantics of our formulas requires that they hold at every circumstance encountered by a (filtering) observer. Other possibilities exist, such as requiring that the stipulations hold at goal states, or even writing extended formulas connecting properties across time. These extensions have yet to be implemented, so are ignored in the remainder of this paper.

### E Seek label map for the finest observer

Previously, we demonstrate the approach to seek label map and plan while disclosing the same plan to the observer. In this section, we will show how to solve \texttt{SEEK}_{x,e} in a similar way.

Firstly, we will show that \( h(W \otimes D) \) is also a finest observer, and will use it in this problem.

**Lemma E-6.** Given world graph \( W \) and divulged plan \( D \), \( h(W \otimes D) \) is a finest observer.

**Proof.** This lemma will be proved by showing that \( \forall s \in \mathcal{L}(W) \cap \mathcal{L}(D), B = \nu_{h(s)}^{h(W \otimes D)}, B' = \nu_{h(s)}^{I} \), such that \( W^{h(W \otimes D), D}_B \subseteq W^{I}_B \).

If there exists an execution \( s \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_{w_1}^W \), such that \( W^{h(W \otimes D), D}_B \) contains only one world state \( w \). Then we also have \( s \in h^{-1}[S_B^{I}] \cap \mathcal{L}(D) \cap S_{w_1}^W \). Hence, \( w_1 \in W^{I}_B \).

When there exists an image execution reaching \( B \) and \( B' \) in \( h(W \otimes D) \) and \( I \), such that \( W^{h(W \otimes D), D}_B \) contains at least two world states \( w_1 \) and \( w_2 \), we have \( \exists s_1 \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_{w_1}^W \), \( \exists s_2 \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_{w_2}^W \). According to Lemma H-7, either \( h(s_1) = h(s_2) \) or \( \nu_{s_1}^W = \nu_{s_2}^W \). Next, we will show that if \( \exists s_1 \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_{w_1}^W \), \( \exists s_2 \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_{w_2}^W \), such that either \( h(s_1) = h(s_2) \) or \( \nu_{s_1}^W = \nu_{s_2}^W \), then \( w_1, w_2 \in W^{I,D}_B \). If \( h(s_1) = h(s_2) \), then it is trivial that \( s_1, s_2 \in h^{-1}[S_B^{I}] \cap \mathcal{L}(D) \cap S_{w_1}^W \), and \( s_2 \in h^{-1}[S_B^{I}] \cap \mathcal{L}(D) \cap S_{w_2}^W \). Hence, \( w_1, w_2 \in W^{I,D}_B \). If \( \nu_{s_1}^W = \nu_{s_2}^W \), then \( s_1, s_2 \in h^{-1}[S_B^{I}] \cap \mathcal{L}(D) \cap S_{w_1}^W \). Hence, \( w_1, w_2 \in W^{I,D}_B \) also holds when at least two world states are consistent with the finest observer’s belief. Hence, \( W^{h(W \otimes D), D}_B \subseteq W^{I,D}_B \) whenever there is one or more estimated world states in \( W^{h(W \otimes D), D}_B \).

**Theorem E-4.** Given any \( s \in \mathcal{L}(W) \cap \mathcal{L}(D) \), the set of I-states reached by \( h(s) \) in \( h(W) \) is \( B = \nu_{h(s)}^{h(W) \otimes D} \). Then \( W^{h(W \otimes D), D}_B = \pi_W(\nu_{h(s)}^{h(W \otimes D)}) \), where \( \pi_W(V) \) takes the first elements from tuples in \( V \) and gives a a set of world states.

**Proof.** According to the definition, \( W^{h(W \otimes D), D}_B = \{ w \in V(h(W \otimes D)) | h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(D) \cap S_w^W \neq \emptyset \} = \pi_W(U_{s \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(W) \cap \mathcal{L}(D)} \nu_{h(s)}^{h(W \otimes D)})). \) Since \( \forall s_1, s_2 \in h^{-1}[S_B^{h(W \otimes D)}] \cap \mathcal{L}(W) \cap \mathcal{L}(D) \), we have either \( \nu_{s_1}^W = \nu_{s_2}^W \) or \( h(s_1) = h(s_2) \). If \( \nu_{s_1}^W = \nu_{s_2}^W \), then need to consider the set of executions whose image are equal and defined as \( s' \). Since \( s' \in S_B^{h(W \otimes D)} \), \( s' \) reaches and reaches only vertices in \( B \). Therefore, \( W^{h(W), D}_B \) is the set of all world states included in \( B \). Hence, \( W^{h(W), D}_B = \pi_W(\nu_{h(s)}^{h(W \otimes D)})). \)
Theorem E-4 enables us to build the estimates for a set of observer I-states $B \subseteq V(h(W))$. By evaluating the stipulations on $\pi_W(B)$, we are able to mark the set observer I-states $B$ as either satfd$(B, \Phi) = True$ or satfd$(B, \Phi) = False$. The label map satisfies the stipulations, iff

$$\forall s \in L(P) \cap L(W), \text{satfd}(B, \Phi), \text{where } B = V_{h(s)}^h(W \otimes D)$$

Hence, we need to find a label map such that the set of vertices in $W \otimes D$, which we call observer's estimate, reached by $P$ always satisfy the stipulations.

Before searching for the label map, we will firstly mark each vertex $v \in V(W \otimes D)$ as reachable if $\exists s \in L(P), v \in V_s^W \otimes D$. This can be done by making a product graph of $W \otimes D$ and $P$. The vertex in $W \otimes D$ is reachable if it is paired with some vertex in $P$ in the product graph. Then for any I-states $B \subseteq W \otimes D$, $B$ is reachable if there exists a vertex $v \in B$ is reachable by $P$.

Next, we will build a search tree to search for the label map. Similar to $\text{SEEK}_{\pi, \lambda}$, we also interprete the label map as a set of partitions. These partitions serve as an important branching factor in the search tree. Starting from $B_0 = V_0(W \otimes D)$, we will expand it as an OR node, where each outgoing edge bearing a partition for all the events at vertices in $B_0$. If the partition conflicts with its ancestors, then we do not expand it. Given a safe partition $P_i = \{X_1, X_2, \ldots\}$, we will treat it as an AND node and each outgoing edge bearing a set of events $X_j \in P_i$. This procedure is exactly as that of Section 6.3. Following the edge with events $X_j$, the states in $B_0$ transition to $B' = \{v' \in V(W \otimes D) | v \in B_0, v' \in X_j, \text{TRANS}(v, v')^W \otimes D\}$. If $B'$ is visited before, then we terminate and do not expand it. Otherwise, we will add it to the stack and expand it later. If $B'$ is reachable by $P$ and violates the stipulations, then we will mark it as stipulation violated.

Then the problem is restated as follows: If there exists a label map, then there exists a subtree by keeping at least one edge for each OR node and all edges for each AND node, such that there is no vertex that violates the stipulations in the subtree.

**F  A set of actions should be chosen for each plan state in $\text{SEEK}_{\pi, \lambda}$**

In $\text{SEEK}_{\pi}$, we only need to choose an action at each plan state. However, this is no longer true in $\text{SEEK}_{\pi, \lambda}$. Since the plan to be sought is also the plan to be divulged, the chosen actions also help to disguise the transitions under the label map. An example is given as follows: In the world graph shown in Fig. F-11, we can just pick either $a_1$ or $a_2$ at $w_1$ in the planning problem. But to solve problem $\text{SEEK}_{\pi, \lambda}$ with stipulation $\Phi = (\neg w_3 | w_4) \wedge (w_3 | \neg w_4)$, we have to choose both action $a_1$ and $a_2$ when reaching $w_1$, and map them to the same image in the label map. Then the observer will never be able to distinguish the transitions to $w_3$ and $w_4$. Hence, it is necessary to choose a set of actions at a particular world state in $\text{SEEK}_{\pi, \lambda}$, when the plan is also disclosed.

**G  Solving Check and Seek problems with CTL**

In this section, we will present solutions of Check and Seek plan by utilizing CTL instead of implementing and-or search in Python. In these solutions, we will firstly build
a Kripke structure to capture the system dynamics, then specify the stipulations and goal conditions as temporal properties that should be satisfied in the computation tree of the Kripke structure. With both Kripke structure and CTL specification, the software nuXmv is able to evaluate whether these properties are satisfied or not, and give a counterexample if the CTL specification does not hold. We will use this mechanism to solve CHECK problem and SEEK plan.

For CHECK, we will firstly transform $W \otimes P \otimes SDE(I)^{-1}$ into a Kripke structure, where each vertex $(v^W, v^P, v^{SDE(I)^{-1}})$ is marked as stipulation satisfied if $v^{SDE(I)^{-1}}$ satisfies the stipulations. Similarly in the problem of SEEK plan, we will also encode $W \otimes D \otimes SDE(I)^{-1}$ into a Kripke structure, where each vertex $(v^W, v^P, v^{SDE(I)^{-1}})$ is marked as a goal state if all world states associated with $v^P$ are in the goal region, and stipulation satisfied if $v^{SDE(I)^{-1}}$ satisfies the stipulations. Then the computation tree of these Kripke structure is shown in the lower half of Fig. G-12. It has two phases: the initialization phase and execution phase. In the initialization phase, every variable, including action, observation, label map, is initialized from its domain. Given a plan and label map, these variables will be initialized a value by the plan and label map. Otherwise, the variable will be assigned a value from its domain. In the execution phase, the system assigns a value for observation, then assigns a value for action at the next time step, according to the Kripke structure. At each time step, only one variable is updated and, as there are several choices for assignments of variables, this gives branches in the computation tree.

CTL introduces two kinds of temporal operators, in addition to the logical operators. The first group is quantifiers over paths:

- $\forall p$ requires that $p$ should be true on all paths starting from the current state.
- $\exists p$ requires that $p$ should be true on at least one path from the current state where $p$ also holds.

The second group is the path-specific quantifiers. Only the ones that will be used in this paper are listed:

- $\forall p$ requires that $p$ should be true at the next state in the path.
- $p \mathbin{\mathcal{U}} q$ requires that $p$ has to be true util $q$ holds. Note that $q$ is also required to hold some time in the future.
To CHECK whether the goal is reached, we will use \(\text{"A} \text{[state is valid } \cup \text{ goal is reached]}\)”, which guarantees: (i.) the plan is safe and correct since all states reached by the plan are valid states; (ii.) the plan is finite and live since the goal will eventually be reached in finite number of steps. Similarly, to specify that the stipulations are satisfied all the way toward goal states, we use CTL \(\text{"A} \text{[stip_satisfied } \cup \text{ (goal is reached } \land \text{ stip_satisfied)]}\”.

To SEEK the plan that reaches the goal and satisfies the stipulations, we will use the combination of \((Ax \ Ext)\). \(Ax\ p\) requires that \(p\) holds at all the children states, which is same as the search in AND nodes. \(Ext\ p\) requires that \(p\) holds at one of the children states, which shares the same spirit as the search for OR nodes. With \(k\) combinations of \(Ax \ Ext\), we can search for the plan with depth \(k\) in terms of the action nodes. Hence, the we can use CTL \(\text{"Ex stip_satisfied } (\land (\text{goal } \lor Ax \text{ stip_satisfied } \land (\text{goal } \lor Ext \text{ stip_satisfied})))^k\) (Goal \(\land\ stap_satisfied))” to search for the plan with maximum depth \(k\). We will increase \(k\) from 0 to \(|V(W)| \times |V(D)| \times V(I)|\), which is the upper bound given in Theorem 1.

Once it has been determined that a plan exists, we can extract a plan by using the counterexamples from the negated CTL. The counterexample of the negated CTL only serves as one execution of a plan, which only gives us the action choices for a part of the plan. But we are still able to get other actions choices in two ways: (i.) By changing the Kripke structure, we can force the system to transition to the other executions and get additional counterexamples. Combing all these counterexamples, we can build the plan. (ii.) By changing the CTL and initial state of the Kripke structure, we can find an action for each action state appeared in the plan. Eventually, we are able to build the whole plan. In this paper, we use the second method and the procedure is shown in Algorithm 5.
H  Lemmas and proofs on observers

Lemma H-7. Three closely related claims:

- For every pair of executions $s_1, s_2 \in h^{-1}S_B^{h(W)} \cap L(D)$, then we have either $h(s_1) = h(s_2)$ or $V_{s_1}^W = V_{s_2}^W$.
- For every pair of executions $s_1, s_2 \in h^{-1}S_B^{h(D)} \cap L(D)$, then we have either $h(s_1) = h(s_2)$ or $V_{s_1}^D = V_{s_2}^D$.
- For every pair of executions $s_1, s_2 \in h^{-1}S^{h(W \otimes D)} \cap L(W) \cap L(D)$, then we have either $h(s_1) = h(s_2)$ or we have $V_{s_1}^W = V_{s_2}^W$ and $V_{s_1}^D = V_{s_2}^D$.

Proof. For $\forall s_1, s_2 \in h^{-1}S_B^{h(W)}$, we have $V_{h(s_1)}^h(W) = V_{h(s_2)}^h(W) = B$. Suppose $h(s_1) \neq h(s_2)$ and $V_{s_1}^W \neq V_{s_2}^W$. Let $w_1 \in V_{s_1}^W$ and $w_1 \notin V_{s_2}^W$. Then we have $w_1 \in V_{h(s_1)}^h(W)$. In order to satisfy $V_{h(s_1)}^h(W) = V_{h(s_2)}^h(W)$, we need to find another execution $s' \in h^{-1}S_B^{h(W)}$ such that $h(s') = h(s_2)$ and $w_1 \in V_{s'}^W$, which contradicts with the condition $s_1, s_2 \in h^{-1}S_B^{h(W)}$.

Similarly, one proves that $\forall s_1, s_2 \in h^{-1}S_B^{h(D)} \cap L(D)$, then we have either $h(s_1) = h(s_2)$ or $V_{s_1}^D = V_{s_2}^D$.

Now, $\forall s_1, s_2 \in h^{-1}S_B^{h(W \otimes D)} \cap L(W) \cap L(D)$, we have $s_1, s_2 \in h^{-1}S_B^{h(W)} \cap L(W)$ and $s_1, s_2 \in h^{-1}S_B^{h(D)} \cap L(D)$. Then we have either $h(s_1) = h(s_2)$ or, otherwise, both $V_{s_1}^W = V_{s_2}^W$ and $V_{s_1}^D = V_{s_2}^D$. \qed

Lemma 2. $h(W)$ is a finest observer.

Proof. This lemma will be proved by showing that $\forall s \in L(W), B = V_{h(s)}^h(W)$ and $B' = V_{h(s')}^h(W)$, we have $V_B^h(W), D \subseteq W_{B'}^h(W)$.

If $V_{h(s)}^h(W), D$ contains only one world state $w$, then $\exists s \in h^{-1}S_B^{h(W)} \cap L(D)$, we also have $s \in h^{-1}S_B^{h(W)} \cap L(D)$, and $\exists s \in S_B^{h(W)} \cap L(D) \cap S_w^W$. Hence, $w$ is also contained in $W_B^h(W)$.

If $V_{h(s')}^h(W), D$ contains at least two world state $w_1, w_2$, then $\exists s_1 \in h^{-1}S_B^{h(W)} \cap L(D)$, $s_1 \in h^{-1}S_B^{h(W)} \cap L(D)$, and $s_2 \in h^{-1}S_B^{h(W)} \cap L(D) \cap S_w^W$. According to Lemma H-7, $h(s_1) = h(s_2)$ or $V_{s_1}^W = V_{s_2}^W$. If $h(s_1) = h(s_2)$, then it is trivial that $s_1, s_2 \in h^{-1}S_B^{h(W)}$. We have $s_1 \in h[S_B^{h(W)} \cap L(D) \cap S_w^W]$, and $s_2 \in h[S_B^{h(W)} \cap L(D) \cap S_w^W]$. Hence, $w_1, w_2 \in W_{B'}^h(W)$. If $V_{s_1}^W = V_{s_2}^W$, then $s_1, s_2 \in h[S_B^{h(W)} \cap L(D) \cap S_w^W]$. Hence, $w_1, w_2 \in W_{B'}^h(W)$ also holds.

Therefore, $V_B^{h(W), D} \subseteq W_{B'}^{h(W), D}$. Hance, $h(W)$ is a finest observer. \qed

I  Theorems, lemmas and proofs on plans

Lemma I-8. Given any plan $(P, V_{\text{term}})$, there exists a plan $(P', V'_{\text{term}})$ that is congruent on the world graph $W$ and $L(P') = L(P)$.
Proof. We give a construction from $P$ of $P'$ as a tree, and show that it meets the conditions. To construct $P'$, perform a BFS on $P$. Starting from $V_0(P)$, build a starting vertex $v_0$ in $P'$, keep a correspondence between it and $V_0(P)$. Mark $v_0$ as unexpanded. Now, for every unexpanded vertex $v$ in $P'$, mark the set of all outgoing labels for its corresponding vertices in $P$ as $L_v$, create a new vertex $v'$ in $P'$ for each label $l \in L_v$, build an edge from $v$ to $v'$ with label $l$ in $P'$, and mark it as expanded. Repeat this process until all vertices in $P'$ have been expanded. Mark the vertices corresponding to vertices in $V_{term}$ as $V'_{term}$. In the new plan $(P', V'_{term})$, no two executions reach the same vertex. That is, $\forall s_1, s_2 \in V(P'), s_1 \not\sim s_2$. Hence, $P'$ is congruent on $W$. In addition, since no new executions are introduced and no executions in $P$ are eliminated during the construction of $P'$, we have $\mathcal{L}(P') = \mathcal{L}(P)$.

Theorem 1. For problem $\text{SEEK}_{\Phi}((W, V_{goal}), \alpha, (I, D), h, \Phi)$, if there exists a solution $(P, V_{term})$, then there exists a solution $(P', V'_{term})$ that is both $c$-bounded and congruent on $W$, where $c = |V(W)| \cdot |V(D)| \cdot |V(I)|$.

Proof. Suppose $\text{SEEK}_{\Phi}$ has a solution $(P, V_{term})$. Then the existence of a solution $(P', V'_{term})$ which is congruent on $W$ is implied by Lemma I-8. Moreover, we have $\text{CHECK}((W, V_{goal}), (P, V_{term}), D, I, h, \Phi) \Rightarrow \text{CHECK}((W, V_{goal}), (P', V'_{term}), D, I, h, \Phi)$, following from two observations:

(i) if $(P, V_{term})$ solves $(W, V_{goal})$ then the means of construction ensures $(P', V'_{term})$ does as well, and

(ii) in checking $\Phi$, the set of estimated world states $W_{(v)}^{I,D}$ does not change for each vertex $v \in V(SDE(I))$, since the triple graph is independent of the plan to be searched.

The set of I-states to be evaluated by $\Phi$ in SDE$(I)$ is $\cup_{v \in hI(P) \cap L(W)} V_{SDE(I)}^{\Phi}$. Since $\mathcal{L}(P) = \mathcal{L}(P')$, the set of I-states to be evaluated is no altered and the truth of $\Phi$ along the plan is preserved.

The final step is to prove that if there exists a congruent solution $(P', V'_{term})$, then there exits a solution $(P'', V''_{term})$ that is $c$-bounded. First, build a product graph $T$ of $W$, $D$, and $h^{-1}(SDE(I))$, with vertex set $V(W) \times V(D) \times V(h^{-1}(SDE(I)))$. Then trace every execution $s$ in $P'$ on $T$. If $s$ visits the same vertex $(v^W, v^D, v^h(SDE(I)))$ multiple times, then $(v^W, v^D, v^{-1}(SDE(I)))$ have to be action vertices, for otherwise $P$ can loop forever and is not a solution (since $P'$ is finite on $W$). Next, record the action taken at the last visit of $(v^W, v^D, v^{-1}(SDE(I)))$ as $a_{last}$. Finally, build a new plan $(P'', V''_{term})$ by bypassing unnecessary transitions on $P'$ as follows. For each vertex $(v^W, v^D, v^{-1}(SDE(I)))$ that is visited multiple times, $P''$ takes action $a_{last}$ when $(v^W, v^D, v^{-1}(SDE(I)))$ is first visited. $P''$ terminates at the goal states without violating any stipulations, since it takes a shortcut in the executions of $P'$ but—crucially—without visiting any new observer-I-states. In addition, $P''$ will visit each vertex in $T$ at most once, and the maximum length of its executions is $|V(W)| \times |V(D)| \times |V(h^{-1}(SDE(I)))|$. Since $P''$ preserves the structure of $P'$ during this construction, $P''$ is also congruent.

Lemma I-9. Let $W$ be estimated world states for the finest observer $h(W)$, and let $w$ be the world state which is observable to the robot. If there exists a solution for $\text{SEEK}_{\alpha,\lambda}$, then there exists a solution that only visits each pair $(w, W)$ at most once.
Proof. Let \((P, V_{\text{term}})\) and \(h\) be a solution for \(\text{EEK}_{w,\lambda}\). Suppose \(P\) visited \((w, W)\) \(n\) times. Let the set of actions taken at \(i\)-th visit be \(A_i\). Then we can construct a new plan \((P', V_{\text{term}})\) which always takes \(A_n\) at \((w, W)\). If \(P\) does not violate the stipulations, then \(P'\) will never do since \(P'\) is a shortcut of \(P\) and never visits more I-states than \(P\) does. In addition, \(P'\) will also terminate at the goal region if \(P\) does. \(\square\)

**Theorem 2.** If there exists a solution for \(\text{SEEK}_{x,\lambda}((W, V_{\text{goal}}), x, (I_f, x), \lambda, \Phi)\), then there exists a plan \(P\) that takes \((w, W_B^{h(W)}P)\) as its plan state, where \(w\) is the world state and the set \(W_B^{h(W)}P\) consists of the estimated world states for I-states \(B\). Furthermore, if \((w, W_B^{h(W)}P) \in V(P)\), then \(\forall w' \in W_B^{h(W)}P, (w', W_B^{h(W)}P) \in V(P)\).

Proof. Lemma 1-9 shows that we can treat \((w, W_B^{h(W)}P)\) as the plan state for the plan to be searched for.

Since \(w' \in W_B^{h(W)}P\), we have \(\exists s \in S_w^W \cap \mathcal{L}(P) \cap h^{-1}[S_B^{h(W)}]\). Since \(s \in \mathcal{L}(P)\), \(s\) reaches \(w\) and \(h(s)\) reaches \(B\), we have \(s\) reaches the tuple \((w', W_B^{h(W)}P)\). Hence, \((w', W_B^{h(W)}P) \in V(P)\). \(\square\)

### J Pseudocode for algorithms

**Algorithm 1: SDE(G)**

1. Build initial vertex \(v_0'\) in \(G'\), associate \(v_0'\) with all \(v_0 \in V_0(G)\): \(A(v_0') \leftarrow V_0(G)\)
2. Initialize queue \(Q \leftarrow \{v_0'\}\)
3. While \(Q\) not empty do
   1. \(s' \leftarrow Q\text{.pop}\)
   2. // Refine each label and determine which states each refinement maps to:
      - \(L \leftarrow\) all outgoing edge labels of \(s'\)
      - \(L' \leftarrow \text{RefineLabels}(L)\) // See Alg.1 in the paper: F. Z. Saberifar, S. Ghasemlou, J. M. O’Kane, and D. A. Shell, “Set-labelled filters and sensor transformations”, RSS, 2016
      - \(d_{\text{Lab}}[\cdot] \leftarrow \emptyset\) // Empty a map
   4. For every outgoing edges of \(s'\), record which states you reach with \(\text{Representative}(l')\) by adding them to \(d_{\text{Lab}}[l']\)
   5. // Produce new states as need:
      - For \((l_a, V_a) \in d_{\text{Lab}}\) do
         1. \(flag \leftarrow\) False
         2. For \(t \in V(G')\) do
            1. If \(V_a = A(t)\) then
               1. Add \(s' \xrightarrow{l_a} t\) in \(G'\)
               2. \(flag \leftarrow\) True
            3. If \(flag =\) False then
               1. Create new state \(t'\), add \(s' \xrightarrow{l_a} t'\) in \(G'\), \(A(t') \leftarrow V_a\)
      6. Return \(G'\)
Algorithm 2: CHECKSOLN($W, V_{\text{goal}}, P, V_{\text{term}}$)

\[
Q \leftarrow [] \text{ and visited } \leftarrow []
\]

for $w \in V_0(W)$ do

for $v \in V_0(P)$ do

if $v \in V_{\text{term}}$ and $w \not\in V_{\text{goal}}$ then

return False // Not correct

else if $v \not\in V_{\text{term}}$ then

$Q$.append(($w, v$))

visited.append(($w, v$))

while $Q$ not empty do

($w, v$) $\leftarrow$ $Q$.pop

$N_w \leftarrow W$.outNeighbors($w$)

$N_v \leftarrow P$.outNeighbors($v$)

if $N_w$ is empty and $N_v$ not empty then

return False // Not safe and not live

if $w$ is action vertex and $N_v$ $\not\subseteq$ $N_w$ then

return False // Not safe on action

else if $w$ is observation vertex and $N_w$ $\not\subseteq$ $N_v$ then

return False // Not safe on observation

for $w \in N_w$ do

for $v \in N_v$ do

if $w \not\in V_{\text{goal}}$ and $v \in V_{\text{term}}$ then

return False // Not correct or live

else if $w \in V_{\text{goal}}$ and $v \in V_{\text{term}}$ then

continue // Terminating vertex

else if ($w, v$) $\in$ visited then

return False // Loop detected, not finite

else

$Q$.append(($w, v$))

visited.append(($w, v$))

return True // Passing all the tests
Algorithm 3: ActionExpand($W, P, h, \Phi$)

if $W \subseteq V_{\text{goal}}$ then
    return $(P, h)$

AllActChoices $\leftarrow \{\}$

for $w \in W$ do
    AllActChoices $\leftarrow$ AllActChoices $\times$ $w$.avblActs // Encode action choices for states in $W$
    cartesian product

for actChoice $\in$ AllActChoices do
    AllChosenActs $\leftarrow \{\}$
    for $w \in W$ do
        $A_w \leftarrow$ actChoice[$w$] // Obtain the set of action choices for each state
        AllChosenActs $\leftarrow$ AllChosenActs $\cup$ $A_w$

    $P[W]$ $\leftarrow$ actChoice // Put action choices in the plan

    AllPartitions $\leftarrow$ All partitions of AllChosenActs

    for partition $\in$ AllPartitions do
        $(P', h')$ as a copy of $(P, h)$
        NoSoln $\leftarrow$ False
        if $h'$ does not conflict with partition then
            $h' \leftarrow h$.integrate(partition)
            for group $\in$ partition do
                $W' \leftarrow$ vertices $W$ transitions to under group
                if $W'$ satisfies stipulation $\Phi \land P$.contains($W'$)=$\text{False}$ then
                    $(P', h') \leftarrow$ ObservationExpand($W'$, $P'$, $h'$, $\Phi$)
                    if $(P', h')$ is empty then
                        NoSoln $\leftarrow$ True
                        break
                else
                    NoSoln $\leftarrow$ True
                    break
        if NoSoln is False then
            return $(P', h')$

return empty
Algorithm 4: ObservationExpand($W, P, h, \Phi$)

\begin{algorithmic}
  \If{$W \subseteq V_{\text{goal}}$}
    \State \Return $(P, h)$
  \EndIf
  \State $\text{AllObs} \leftarrow \{\}$
  \For{$w \in W$}
    \State $O_w \leftarrow w.\text{avblObs}$
    \State $\text{AllObs} \leftarrow \text{AllObs} \cup O_w$
    \State $P'[w] \leftarrow \text{AllObs}$
    \State $\text{AllPartitions} \leftarrow \text{All partitions of AllObs}$
    \For{$\text{partition} \in \text{AllPartitions}$}
      \State $(P', h')$ as a copy of $(P, h)$
      \State $\text{NoSoln} \leftarrow \text{False}$
      \If{$h'$ does not conflict with partition}
        \For{$\text{group} \in \text{partition}$}
          \State $W' \leftarrow \text{vertices } W \text{ transitions to under group}$
          \If{$W'$ satisfies stipulation $\Phi \land P.\text{contains}(W')=\text{False}$}
            \State $(P', h') \leftarrow \text{ActionExpand}(W', P', h', \Phi)$
          \EndIf
          \If{$(P', h')$ is empty}
            \State $\text{NoSoln} \leftarrow \text{True}$
            \State \Break
          \EndIf
        \EndFor
        \Else
          \State $\text{NoSoln} \leftarrow \text{True}$
        \EndIf
      \EndIf
    \EndFor
  \EndFor
  \If{$\text{NoSoln}$ is \text{False}}
    \State \Return $(P', h')$
  \EndIf
  \State \Return empty
\end{algorithmic}
Algorithm 5: BUILDPLAN(W, CTL, initState)

NCLT←“!("+CTL+)")”
(result, counterexample)← nuXmv.evaluate(W, NCLT, initState)
if result is False then
  (s, event, t)← counterexample.next()
  initialize plan as empty
  if event is an action and s ∉ V\textsubscript{goal} then
    CTL.removeFirstEX()
    subplan← BuildPlan(W, CTL, t)
    plan.addTransition(s, event, subplan.initState)
  else if event is an observation and s ∉ V\textsubscript{goal} then
    CTL.removeFirstAX()
    for obs ∈ W.outEvents(s) do
      tgt ← W.transTo(s, obs)
      subplan← BuildPlan(W, CTL, tgt)
      plan.addTrans(s, obs, subplan.initState)
    else
      plan.addTermVertex(s)
  plan.initState← initState
return plan
else
  return empty