Which Actions Are Always Necessary in Fully-Observable Non-Deterministic Planning?

Yaofang Zhang and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
vzhang@cs.ucla.edu, darwiche@cs.ucla.edu

Abstract

Explaining the sequential action choices of an autonomous agent is essential for understanding the agent and learning from the agent. Given a goal-reaching action sequence in a non-deterministic environment, one important aspect of explaining action choices is deciding if the action choice is always necessary for the action sequence, or if it may be omitted under certain circumstances. Most works in explainable AI planning (XAIP) have been focused on the explanation of action choices in deterministic environments as assumed by classical planning, and one widely used notion is the *causal link*. In this work, we will apply this notion of causal link to justifying the action choices of a trace in fully-observable non-deterministic (FOND) settings. We will also introduce a new notion – the *always-necessary actions* of a trace – and propose an approach that derives the always-necessary actions of a trace from the causal link justifications of action choices.

1 Introduction

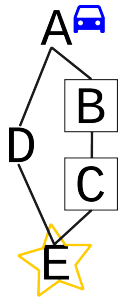
Suppose a student is watching a surgeon removing a tumor, and the student is interested in why a certain incision is performed. The incision in question may be always necessary, or it may be only necessary in this particular operation (e.g., because of some complication that occurred). It can be helpful for the student to know the set of actions that are always necessary for the operation versus those that were done because of the particular circumstance. In this paper, we propose an approach that considers a sequence of actions that achieve the intended goal, called a goal-reaching trace, in the fully-observable non-deterministic (FOND) planning setting, and that identifies the set of always-necessary actions in the trace.

Consider a scenario (Figure 1a) in Tyreworld – a classic FOND domain. An agent (a car in this example) is initially at A, and the goal is to arrive at E. The agent can move to a connected adjacent location, but doing so might (i.e., non-deterministically) cause a flat tire. Moving to a new location is not possible with a flat tire, and only certain locations (B and C in this example) have a spare tire. Suppose the agent performed the following sequence of five actions (labeled $a_0 - a_4$ in Figure 1b): move from A to B, pick up a spare tire at B, change tire, move from B to C, and move from C to E. In this circumstance, moving from A to B happened to cause a flat tire, and neither of the two later move actions led to a flat tire. Which actions in this sequence may be omitted under some circumstances? Which actions in the sequence are always necessary in all possible circumstances that the agent may experience while following this action sequence?

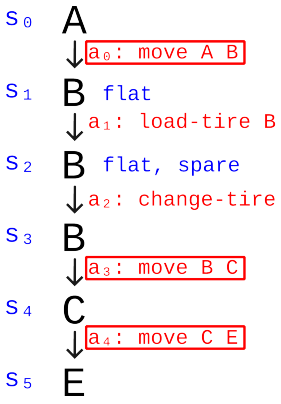
For this particular circumstance, none of the five actions can be omitted. For example, if changing the tire is omitted, the action of moving from B to C is not applicable because of the flat tire, and it is not possible to reach the goal with the remaining action sequence (or any sub-sequence). Recall, however, that moving from one location to an adjacent one may non-deterministically cause a flat tire. If moving from A to B did not cause a flat tire, the agent did not need to pick up a spare tire at B and change the flat tire, so some actions can be omitted. We say in this case that we have a different circumstance since the environment transitions are different.

We will call actions that can never be omitted under any circumstances *always-necessary actions*. In our example, these actions are indicated with boxes in Figure 1b: move from A to B, move from B to C, and move from C to E. If any one of them is not performed (e.g., moving from B to C), the remaining action sequence (or any subset of it) can never achieve the goal. Two of the actions in the sequence are not always necessary: picking up the spare tire at B, and changing out the flat tire. That is, under certain circumstances, the agent can omit these actions from the action sequence, and achieve the goal with only the always-necessary actions.

Our main contribution in this paper is that for a goal-reaching trace (i.e., a sequence of state-action pairs) in a FOND setting, we determine whether each action in the sequence is necessary to reach the goal under all possible circumstances that may be encountered while executing the action sequence, including the particular circumstance encountered in the trace. We show that the always-necessary actions of a trace can be constructed without access to a planner, or knowledge about how the agent came up with the action sequence. Our approach builds on the notion of *causal link explanation* used by prior work in the explainable AI planning (XAIP) field (Seegebarth et al. 2012; Sreedharan, Muise, and Kambhampati 2022) but with some



(a) An example of a setting in the Tyreworld domain.



(b) A goal-reaching trace.

Figure 1: (a) We have 5 locations, A-E, each represented as a node in the graph. The roads connecting locations are represented as edges. The square boxes encircling B and C denote that each of those two locations has a spare tire. In the initial state, the vehicle is at A, has no flat tire, and carries no spare tire. The goal is to move to E. (b) A goal-reaching trace that has six states (s_0 to s_5) and five actions (a_0 to a_4). For each state, the values of the three key state variables are shown: the location of the vehicle, whether the vehicle has a flat tire (denoted by presence "flat") or no flat tire, and whether the vehicle carries a spare tire (denoted by the presence "spare") or no spare. The always-necessary actions (a_0, a_3, and a_4) are indicated by the boxes around them.

important differences that we discuss later.

This paper is organized as follows. Section 2 introduces fully-observable non-deterministic (FOND) planning. Section 3 proposes an approach for finding the always-necessary actions of a goal-reaching trace in FOND planning, and Section 4 demonstrates the approach using more concrete examples. Section 5 closes with some concluding remarks.

2 Background

In this section, we will introduce the FOND planning problem and ground it with our Tyreworld example.

2.1 FOND Planning Problem

A fully-observable non-deterministic (FOND) planning problem can be represented by a tuple $P = \langle \mathcal{X}, \mathcal{A}, s_0, g \rangle$, where $\mathcal{X}$ is the set of discrete state variables, $\mathcal{A}$ is the set of actions, s_0 is the initial state, and g is the goal.

State A complete assignment of the state variables represents a *state* of the world. A set of states can be represented by a logical formula (for each state in the set, the corresponding variable assignment satisfies the logical formula). The initial state s_0 is a complete assignment of the state variables. The goal g is a formula over state variables.

Action An action a is represented by its precondition $\texttt{precond}(a)$, and its effects $\texttt{eff}(a)$. Action a's precondition $\texttt{precond}(a)$ is represented by a logical formula which characterizes the set of states for which the action is applicable. Action a's effects $\texttt{eff}(a)$ is a set of possible effects. The set of possible next states of applying an action a at state is denoted by $\texttt{apply}(s, a)$. Each possible effect is represented by a term; the variables that appear in the term take on the newly assigned value, and variables that are not in the term remain unchanged. Recall that a *term* is a conjunction of variable settings (represented as a set of variable settings). A variable setting is called a *literal*. Finally, $\alpha \models \beta$ means that formula α logically implies/entails formula β. That is to say, α is stronger than β, or β is weaker than α.

In the FOND planning field, the SAS^+ formulation (Bäckström and Nebel 1995) is widely adopted, thanks to the PDDL-to-SAS^+ translation method (Helmert 2009). In FOND SAS^+, $\texttt{precond}(a)$ is always a term (over the state variables), which is called "partial state" in (Helmert 2009; Muise, McIlraith, and Beck 2012). To simplify notation, our formulation here does not place such restrictions on $\texttt{precond}(a)$.

Trace The notion of a trace is central to our formulation.

Definition 1 (Trace)**.** *A trace τ is a sequence of state-action pairs $[(s_0, a_0), (s_1, a_1), \ldots, (s_n,)]$, such that $s_i \models precond(a_i)$ and $s_{i+1} \models apply(s_i, a_i)$. Trace τ is goal-reaching iff $s_n \models g$.*

The notion of a "trace circumstance" that we used informally earlier corresponds to the state sequence (transition) $s_0, s_1, \ldots, s_n$. Since FOND is non-deterministic, it is possible to have two traces with the same initial state, the same action sequence, and that achieve the same goal while having two different circumstances (i.e., state transitions).

2.2 Tyreworld Example

Let us revisit the Tyreworld example. We will introduce the following state variables for the problem setting shown in Figure 1a. A Boolean variable F represents whether the vehicle has a flat tire (f) or no flat tire (f). Similarly, another Boolean variable S represents whether the vehicle carries a spare tire (s) or no spare (s). For locations B and C, we will have boolean variables B and C to represent whether the location has a tire (b, c) or no longer has a tire (b, c). We will not introduce such variables for the other three locations because those three locations never have a tire. Finally, a discrete variable L represents the location of the vehicle, with literals l_A, l_B, l_C, l_D, l_E representing vehicle at location A-E respectively.

The goal-reaching trace shown in Figure 1b can be represented as follows:

- Initial state $s_0 = \{f, s, l_A, b, c\}$. Action a_0 (moving from A to B) has precondition $f \wedge l_A$, and two possible effects $\{l_B\}$ and $\{f, l_B\}$ (i.e., the tire unfortunately becomes flat during the move).
- $s_1 = \{f, s, l_B, b, c\}$. Action a_1 (picking up spare tire at B) has precondition $l_B \wedge b$, and only one possible effect $\{s, b\}$.
- $s_2 = \{f, s, l_B, b, c\}$. Action a_2 (changing tire) has precondition s, and two possible effects $\{\}$ (i.e., nothing happens) and $\{f, s\}$ (i.e., the tire is changed successfully).
- $s_3 = \{f, s, l_B, b, c\}$. Action a_3 (moving from B to C) has precondition $f \wedge l_B$, and two possible effects $\{l_C\}$ and $\{f, l_C\}$.

– $s_4 = \{f, s, l_C, b, c\}$. Action a_4 (moving from C to E) has precondition $f \wedge l_C$, and two possible effects $\{l_E\}$ and $\{f, l_E\}$.
– $s_5 = \{f, s, l_E, b, c\}$, which satisfies the goal l_E.

3 Always-Necessary Actions of a Goal-Reaching Trace

The main goal of this section is to define the notion of an *always-necessary action set* (ANAC) given a goal-reaching trace τ. Intuitively, ANACs help in answering the following question: Starting with the initial state of trace τ, can we achieve the goal using only a sub-sequence of the actions in trace τ? An ANAC S tells us that any such sub-sequence must include at least one action from the set S. Moreover, this is guaranteed to hold regardless of which circumstance (i.e., state transition) may materialize due to non-determinism when executing the sub-sequence. This also explains why members of the set S are called always-necessary, as opposed to necessary, since the necessity of one of these actions S is independent of the circumstance underlying a trace. We further remark that a trace may lead to multiple ANACs so the interest is in finding all of them.

Our approach to ANACs is based on a few notions: action justification (Section 3.1), necessary action (Section 3.2), and justification graph (Section 3.3), which finally lead us to always-necessary actions (Section 3.4).

3.1 Justifying Actions in a Trace

An action is normally chosen because of the changes it causes (i.e., its effects), and as a result, the next states that *may* be reached. In particular, an action is chosen because at least some part of its effect could contribute to either satisfying the goal, or enabling some later action that contributes to the goal. Within a sequence of actions, a particular action a_i can be *justified* by a later action a_j if there is some logical formula α_{ij} (i.e., condition) that satisfies the following three criteria: (1) α_{ij} could be satisfied by executing a_i, (2) α_{ij} is part of the precondition for action a_j, and (3) α_{ij} is not contradicted (i.e., de-established) by the effects of the intermediate actions between actions a_i and a_j. This notion of justification is not new. It is known as a *causal-link explanation* in recent XAIP works (Seegebarth et al. 2012; Sreedharan, Muise, and Kambhampati 2022), where the formula α_{ij} is called a *causal link.*[1] Actions justified in this fashion are called *backward justified.*[2] Note, however, that the formula α_{ij} may already be satisfied by some action that appears before a_i in the sequence. This leads to the notion of a *necessary* action, which has a further requirement that the removal of the action will cause the action sequence to fail. We deviate from prior work on causal-link justifications in the following ways. First, we operate in a non-deterministic setting while almost all prior work treated causal-link justifications in deterministic settings.[3] Next, we treat a more general form of causal links (formula α_{ij}) instead of restricting it to be a single literal (i.e., a variable setting) as done earlier. Finally, we are interested in finding all causal-link justifications instead of just some justifications as done earlier.[4]

We next formally define our generalized notion of a causal (link) justification of an action in the context of a trace. To simplify our formulation, we assume that a dummy action is added to end of the action sequence. The dummy action's precondition is precisely the goal, and its effect is empty.

Definition 2 (Causal Justification). *Let $P = \langle \mathcal{X}, \mathcal{A}, s_0, g \rangle$ be a FOND problem and $\tau = [(s_0, a_0), \ldots, (s_n, a_n)]$ be a goal-reaching trace ($s_n \models g$), where a_n is a dummy action. Let β_i be a logical formula that characterizes the set of all possible next states after applying action a_i in state s_i. A causal justification for action a_i in trace τ is a weakest (i.e., least constrained) formula α_{ij}, $i < j$ such that: $\beta_i \wedge \alpha_{ij}$ is consistent; $precond(a_j) \models \alpha_{ij}$; for all intermediate states $s_l, i < l < j$, $s_l \wedge \alpha_{ij}$ is consistent; and for all earlier state $s_h, h < i$, $s_h \not\models \alpha_{ij}$.*

An action a_i might have no casual justifications. It may also have multiple casual justifications, in which case each will involve a distinct, later action a_j. Note also that a causal justification depends not only on the sequence of actions in a trace but also on its underlying circumstance (i.e., sequence of states in the trace). Hence, this notion applies to a trace but not to plan. Moreover, since FOND planning is non-deterministic, an action's causal justification might not be actually achieved by the action in the given trace.

Coming back to our Tyreworld example trace (Figure 1b), let us consider the causal justifications for each action.

– Action a_0 (move from A to B) has multiple justifications. One is $\alpha_{01} = l_B \wedge b$. Action a_0 fulfills the precondition of the immediate next action a_1 (pick up the spare tire at B). A part of α_{01}, b (location B has a spare tire), is already true in some prior state (the initial state s_0 in this case); the effect of a_0 fulfills the other part of α_{01}, which is l_B (vehicle at location B). The other justification of a_0 is $\alpha_{03} = l_B \wedge f$. Note that, because tire might become flat (i.e., f), α_{03} is not guaranteed to be achieved by a_0. Although α_{03} is actually not achieved in this trace, it is still part of the justification for a_0.
– Action a_1 (pick up the spare tire at B) is only trivially justified by the immediate next action a_2 (change out flat

[1] The causal link is actually the triplet $(a_i; \alpha_{ij}; a_j)$ but since we can recover it from the subscripts of α_{ij} we use the latter notation.

[2] The related notion of *causal landmark* is influential in the broader planning literature as well (Zhu and Givan 2003; Keyder, Richter, and Helmert 2010). Earlier works in the literature also established other notions of justifications for actions; see, e.g., (Fink and Yang 1993; Kambhampati 1995).

[3] In a deterministic setting, the state sequence is unique given an initial state and a sequence of actions. In a non-deterministic setting, the state sequence is not unique and the causal-link justification depends on the specific state sequence of the given trace.

[4] For treatment of causal-link justification in FOND planning, we are only aware of one prior work (Sreedharan, Muise, and Kambhampati 2022) that justifies an action choice $(s; a)$ in a policy by a causal link between the effect of the action a and a policy causal landmark; a policy causal landmark is a fact that is part of an action's precondition, and also must be satisfied by all traces sampled from a policy starting from state s. The causal link and the causal landmark are each a single literal in this case (i.e., a variable setting), and they do not try to discover all causal links.

tire with spare); the justification is $\alpha_{12} = s$.
- Action a_2 is justified by $\alpha_{23} = l_B \wedge f$.
- Action a_3 (move from B to C) is justified by $\alpha_{34} = f \wedge l_C$.
- Action a_4 (move from C to E) is justified by $\alpha_{45} = l_E$, which is the goal.

3.2 Necessary Actions

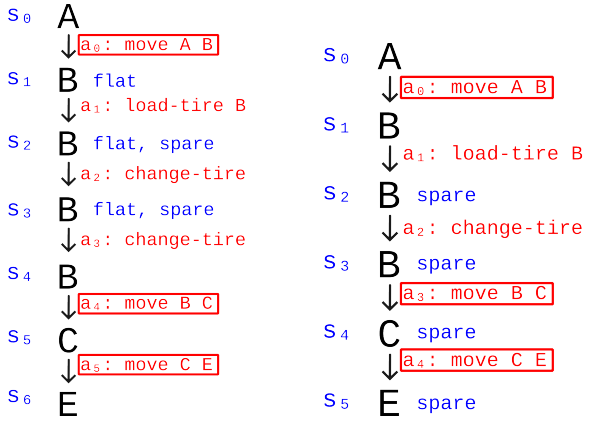
We formalized earlier the causal justifications of an action a_i in the context of a trace τ. Intuitively, a justification is a condition that the action may establish, and its establishment contributes to satisfying either the goal or the precondition of another action a_j that comes later in the trace. An action may have no justifications, in which case it is clearly unnecessary so can be skipped without undermining the goal. However, even an action that has justifications might be unnecessary since the justification α_{ij} may relate to a later action a_j that does not contribute to the goal. We will next formalize the notion of necessary actions.

Definition 3 (Necessary Action)**.** *Let $P = \langle \mathcal{X}, \mathcal{A}, s_0, g \rangle$ be a FOND problem, and $\tau = [(s_0, a_0), \ldots, (s_n, a_n)]$ be a goal-reaching trace ($s_n \models g$), where a_n is a dummy action that has the goal g as its precondition and an empty effect. The necessary actions for the trace τ are defined inductively. The dummy action a_n is necessary. An action a_i, $i < n$, is necessary iff there exists a causal justification α_{ij} such that action a_j ($i < j$) is necessary.*

In other words, an action a_i is necessary if a_i can be justified by a necessary action a_j that comes after a_i in the trace. Hence, the necessity of an action depends on the necessity of *later* actions, but not the other way around. The dummy action a_n is the base case for the inductive definition; an action a_i that has a causal justification α_{in} contributes to satisfying a_n's precondition (i.e., the goal). If action a_i is *unnecessary*, then action a_i either (1) has no justification, or (2) for every justification α_{ij}, a_j is an unnecessary action.

For the trace shown in Figure 1b, all actions are necessary. The last action, a_4 (move from C to E), is necessary because it is justified by the dummy action a_5, and $\alpha_{45} = l_E$ is exactly the goal. Because a_4 is necessary and a_3 (move from B to C) is justified by $\alpha_{34} = f \wedge l_C$, a_3 is necessary. Furthermore, because of justifications α_{03} and α_{23}, actions a_0 and a_2 are both necessary. Finally, action a_1 is necessary because of the justification α_{12} and the necessity of a_2.

Because an action's possible effects are considered for the action's justification, an action is necessary as long as one of the possible effects contributes to the goal, even if that particular effect is not materialized. Consider the goal-reaching trace shown in Figure 2a, the action that changes out the flat tire with the spare is executed twice (a_2 and a_3). The materialized effect of the first execution (a_2) is an empty effect $\{\}$ (the world state is not changed at all). On the second execution (a_3), the materialized effect is $\{f, s\}$ (the flat tire is successfully changed out). Despite action a_2's failure to contribute to the goal, a_2 is not unnecessary. Action a_2 is justified by $\alpha_{24} = l_B \wedge f$, and action a_4 (move from B to C) is necessary. Because at least one of a_2's possible effects contributes to the goal, a_2 could (though not guaranteed to) contribute to the goal and is thus necessary.



(a) An alternative trace with two change-tire actions.

(b) Another trace. There is no flat tire after moving from A to B.

Figure 2: (a) Compared to the trace shown in Figure 1b, the only difference is that, in this trace, action a_2 (change out flat tire with the spare) fails to change the tire (one of this action's possible effects is an empty effect). Action a_3 performs tire changing again, and the flat tire was successfully changed out on this second attempt. (b) This trace has the exact same action sequence as the trace shown in Figure 1b, but in this trace, the tire is not flat after moving from A to B.

We stress that the notion of a necessary action is defined in the context of a trace. For two traces that share the same action sequence but differ in the circumstance, an action can be necessary in one trace but unnecessary in the other. Consider the trace shown in Figure 2b. This trace shares the same action sequence as the trace shown in Figure 1b, but their state sequences differ. In this trace, there is no flat tire after moving from A to B, so the precondition of a_3 (move from B to C) is already satisfied at s_1. Actions a_1 (load spare tire) and a_2 (change out flat tire) are both unnecessary.

3.3 Causal Justification Graphs

Using the notion of causal justification, we can build a graph that represents all the justifications of the actions in a trace. Figure 3 is an example of a *causal justification graph*. Each action's precondition is a node in the graph. To simplify the notation, we will use p_i to denote the precondition of action a_i and also the corresponding node in the graph. Each causal justification α_{ij} corresponds to an edge in the graph from p_i to p_j. For instance, the edge $p_0 \rightarrow p_1$ shows that action a_0 is justified by action a_1 and $\alpha_{01} = p_1$. By definition of a causal justification, $p_j \models \alpha_{ij}$. For this particular example,

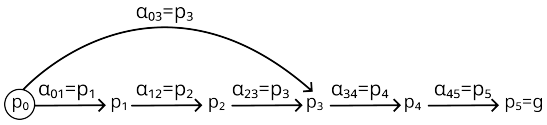


Figure 3: Causal justification graph of the trace in Figure 1b. For each action a_i, node $p_i = precond(a_i)$. We have $p_0 = l_A \wedge f$; $g_1 = l_B \wedge b$; $p_2 = s$; $p_3 = l_B \wedge f$; $p_4 = f \wedge l_C$. Precondition of the dummy action is the goal, $p_5 = g = l_E$. Node p_0 (circled) is satisfied in the initial state.

α_{ij} always equal to p_j, but the equivalence does not always hold, as we show in another example in Section 4.

The causal justification graph is a directed acyclic graph (DAG) because, by definition, the justification of some action a_i can never involve an earlier action a_h (where $h < i$) in the trace. The *source nodes* of this DAG are the set of p_i nodes (i.e., precondition of actions) where p_i is satisfied in the initial state (i.e., $s_0 \models p_i$). The DAG may have multiple sinks but only one of them, the *goal-sink,* satisfies the goal (i.e., precondition of the dummy action). The causal justification graph is a key structure in our following discussions. We will first use it to identify the necessary actions in a trace.

Proposition 1. *Let $P = \langle \mathcal{X}, \mathcal{A}, s_0, g \rangle$ be a FOND problem and $\tau = [(s_0, a_0), \ldots, (s_n, a_n)]$ be a goal-reaching trace ($s_n \models g$). An action a_i is necessary in τ iff the following holds in the justification graph of trace τ: the precondition node p_i of action a_i is on at least one path from a source node to the goal-sink.*

Proof. If the condition is not met, node p_i is either (1) a sink which means that action a_i has no causal justifications, or (2) every path that starts from node p_i terminates in a sink that is not the goal-sink. a_i is unnecessary in both cases. □

3.4 Always-Necessary Action Sets

We are now ready to define the central notion in this paper.

Definition 4 (Always-Necessary Action Set)**.** *Consider a FOND problem $P = \langle \mathcal{X}, \mathcal{A}, s_0, g \rangle$. Let $\tau = [(s_0, a_0), \ldots, (s_n, a_n)]$ be a goal-reaching trace, and S be a subset of its necessary actions. S is an always-necessary action set (ANAC) iff every trace that starts with state s_0, achieves goal g, and whose actions are necessary in τ must include at least one action from S.*[5]

In other words, S is an always-necessary action set if the goal cannot be achieved if we skip all actions in S — regardless of what circumstance may materialize due to nondeterminism. In general, an ANAC S contains multiple actions which means that for a sub-sequence of $[a_0, \ldots, a_{n-1}]$ to have any possibility of reaching the goal, at least one of the actions in S must be included in the sub-sequence. If S contains only one action, then that action is an *always-necessary action*. Note also that a trace may have multiple ANACs in which case we cannot achieve the goal without including at least one action from each ANAC.

There are three ANACs for the trace shown in Figure 3: $\{a_0\}$ (move from A to B), $\{a_3\}$ (move from B to C), and $\{a_4\}$ (move from C to E). Since each is a singleton, each corresponds to an always-necessary action (Section 4 will illustrate ANACs with multiple actions). Without any of these three actions, it is impossible to reach the goal with the rest of the actions in the plan, regardless of what state transitions the agent may experience. Note that action a_1 (pick up tire at B) is not in any of the ANACs. If moving from A to B did not cause a flat tire, the agent can achieve the goal without performing a_1.

[5]Note that a_i and a_j, $i \neq j$, may refer to the same action but at different positions in the action sequence (i.e., repeated actions).

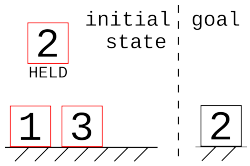


Figure 4: The 3 blocks in this problem are denoted by 1, 2, and 3. In the initial state, 1 and 3 are on the table, while 2 is being held. 1, 2, and 3 are all hot and not destroyed. In the following figures, we will use red outline for a hot block and blue outline for a cold block. Solid outline means the block is not destroyed and dashed outline means it is destroyed. The table is not destroyed in the initial state. The goal is to have 2 on the table, and keep the table not destroyed. States of 1 and 3 are irrelevant for the goal.

We now show how to compute ANACs based on a causal justification graph G using the following result. Recall first that if $p_i \rightarrow p_j$ is an edge in G then p_i is the precondition of action a_i; p_j is the precondition of action a_j; and the edge label α_{ij} is a causal justification for action a_i.

Proposition 2. *Let G be a causal justification graph with only one sink node, the goal-sink.*[6] *Let c be a cut in G that separates the source nodes from the goal-sink and let $S = \{a_i | \alpha_{ij} \in c\}$. Then S is an always-necessary action set.*

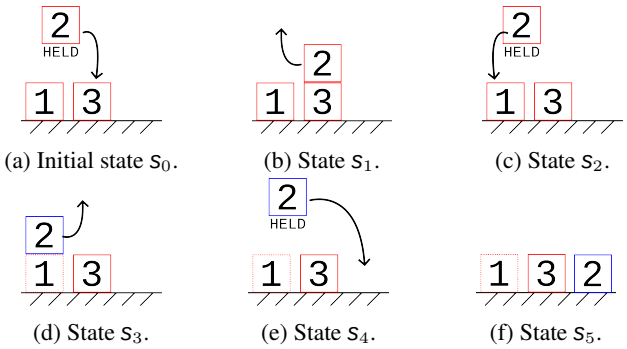
Proof Sketch. Let τ be the trace that generated the causal justification graph G. The nodes of G correspond to the necessary actions in τ. Cut c separates all sources nodes from the one sink node (the goal-sink) in G. Any path from a source node to the goal-sink must include a node in c, which corresponds to an action from S. □

4 Examples in Exploding Blocksworld

To further illustrate the notions introduced in Section 3, we will look at a more complex example — a scenario from the classic Exploding Blocksworld domain as shown in Figure 4. In this domain, a block can be on the table, on another block, or held. There are two types of actions: put the currently held block on top of another block or the table, and pick up a block that does not have another block on top of it. If the currently held block is *hot*, putting it down might (i.e., non-deterministically) cause it to explode, which would destroy the object directly underneath it (another block or the table). Once a block explodes, it becomes *cold*. Once an object is *destroyed*, no new object can be put on top of it. A hot block would not explode if it stays at its current location and is not moved.

There are 3 blocks in the problem shown in Figure 4, which we will denote as 1,2, and 3. We will first introduce the following minimum set of state variables to facilitate later discussion. A discrete variable X represents the location of 2, with states x_1, x_3, and x_t representing 2 on top of 1, 3, and the table, respectively; another state x_0 represents that 2 is held. A Boolean variable $D1$ represents if 1

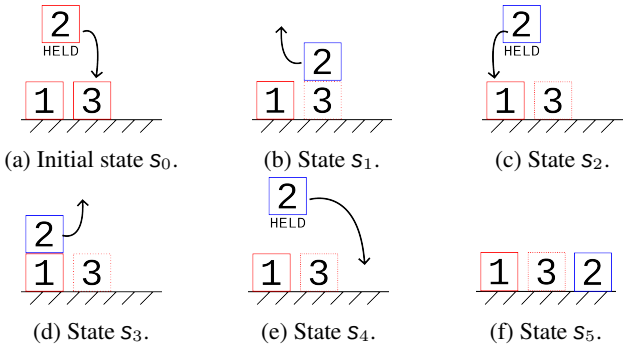
[6]If G does not satisfy this property, we can successively remove sinks that do not correspond to the goal. This would remove nodes p_i where the corresponding action a_i is unnecessary.



(a) Initial state s_0. (b) State s_1. (c) State s_2.

(d) State s_3. (e) State s_4. (f) State s_5.

Figure 5: Trace 1. For each state s_i, the action a_i is illustrated by the black arrow. State s_5 satisfies the goal.



(a) Initial state s_0. (b) State s_1. (c) State s_2.

(d) State s_3. (e) State s_4. (f) State s_5.

Figure 6: Trace 2. The initial state and action sequence are as in Trace 1 (Figure 5) but the state sequence is different.

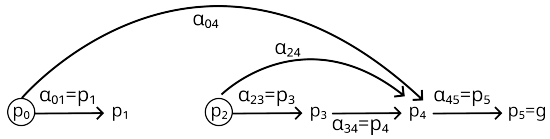
is destroyed ($d1$) or not ($d1$). Similarly, a Boolean variable $D3$ represents if 3 is destroyed ($d3$) or not ($d3$). A Boolean variable H represents if 2 is hot (h) or cold (h). In the initial state, we have x_0, $d1$, $d3$, and h.

4.1 Two Example Traces

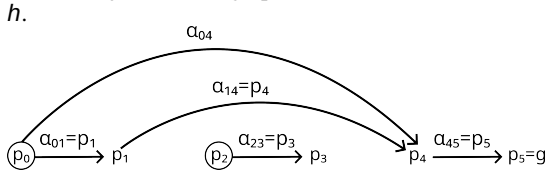
Since we cannot "repair" a destroyed object and keeping the table not destroyed is part of the goal, one strategy of reaching the goal is to repeatedly put the hot 2 on top of either 1 or 3, until 2 explodes and becomes cold; then, putting the cold 2 on the table achieves the goal.

Consider two goal-reaching traces that both adopt this strategy: Trace 1 shown in Figure 5, and Trace 2 shown in Figure 6. The two traces share the same initial state and action sequence, and differ only in the state sequence. In both traces, the following sequence of actions are applied:

- Action a_0 (place 2 on 3) requires 2 to be held (x_0) and 3 not destroyed ($d3$). Also, there cannot be any other block on top of 3, but for conciseness, we will omit mentioning this requirement in the rest of the discussion. a_0 has two possible effects: $\{x_3\}$ and $\{x_3, d_3, h\}$, that is, 2 will be on 3, and 2 *might* explode, which causes 2 to be cold and 3 to be destroyed.
- Action a_1 (pick up 2 from 3) requires 2 on 3 (x_3), and has the effect of 2 being held (x_0).
- Action a_2 (place 2 on 1) and a_3 (pick up 2 from 1) are similar to to a_0 and a_1. We will omit detailed description of their preconditions and effects for brevity.



(a) Causal justification graph for Trace 1. $\alpha_{04} = \alpha_{24} = h$.



(b) Causal justification graph for Trace 2. $\alpha_{04} = h$.

Figure 7: Causal justification graph for the two traces shown in Figures 5 and 6. Source nodes are circled.

- Action a_4 (place *cold* 2 on the table) requires 2 to be held (x_0) and also cold (h). This fulfills the goal of having 2 on the table (x_t).

4.2 ANACs for the Two Traces

For Trace 1 (Figure 5), 2 did not explode when placed on 3, and only exploded when it was later placed on 1. In contrast, in Trace 2 (Figure 6), 2 exploded and became cold when it was placed on 3, so the later action of placing 2 on 1 was unnecessary.

Because of the different circumstances, the justification graphs for the two traces are different. Figure 7a shows the justification graph for Trace 1, which has necessary actions a_0, a_2, a_3, and a_4. There are three cuts of the graph, corresponding to three ANACs for Trace 1 $\{a_0, a_2\}$, $\{a_0, a_2, a_3\}$, and $\{a_4\}$. The second one is subsumed by the first one. This first ANAC shows that at least one of a_0 (placing 2 on 3) and a_2 (placing 2 on 1) should be included in the sub-sequence. The third ANAC shows an always-necessary action a_4 (place the cold 2 on the table).

Figure 7b shows the justification graph for Trace 2, which has necessary actions a_0, a_1, and a_4. The three cuts in the graph correspond to three ANACs: $\{a_0\}$, $\{a_0, a_1\}$, and $\{a_4\}$. The second one is subsumed by the first. Both of the remaining ANACs are singletons, so we have two always-necessary actions: a_0 (place 2 on 3) and a_4 (place cold 2 on the table).

5 Conclusion

For a goal-reaching action sequence in a non-deterministic environment, one important question is whether the action choices are always necessary for achieving the goal, or some may be omitted under certain circumstances. We proposed an approach for addressing this problem which is based on generalizing some prior work on causal-link justifications. Our approach utilized the notion of an action being necessary for a trace, and the notion of a justification graph for a trace, which led to formalizing always-necessary actions as cuts in such a graph. We provided ample concrete examples which illustrate the merits of the proposed formulation of this fundamental notion, the always-necessary actions.