Limitations of Acyclic Causal Graphs for Planning

Anders Jonsson
Dept. Information and Communication Technologies
Universitat Pompeu Fabra
Roc Boronat 138
08018 Barcelona, Spain

Peter Jonsson, Tomas Lööw
Department of Computer Science
Linköping University
SE-581 83 Linköping, Sweden

Abstract
Causal graphs are widely used in planning to capture the internal structure of planning instances. Researchers have paid special attention to the subclass of planning instances with acyclic causal graphs, which in the past have been exploited to generate hierarchical plans, to compute heuristics, and to identify classes of planning instances that are easy to solve. This naturally raises the question of whether planning is easier when the causal graph is acyclic.

In this paper we show that the answer to this question is no, proving that in the worst case, the problem of plan existence is \textbf{PSPACE}-complete even when the causal graph is acyclic. Since the variables of the planning instances in our reduction are propositional, this result applies to STRIPS planning with negative preconditions. We show that the reduction still holds...
if we restrict actions to have at most two preconditions.

Having established that planning is hard for acyclic causal graphs, we study two subclasses of planning instances with acyclic causal graphs. One such subclass is described by propositional variables that are either irreversible or symmetrically reversible. Another subclass is described by variables with strongly connected domain transition graphs. In both cases, plan existence is bounded away from $\text{PSPACE}$, but in the latter case, the problem of bounded plan existence is hard, implying that optimal planning is significantly harder than satisficing planning for this class.

**Keywords:** planning, computational complexity

### 1. Introduction

The causal graph offers insight into the interdependence among the variables of a planning instance. A sparse causal graph characterizes a planning instance with few variable dependencies, potentially making it easier to determine when and how to change the value of some variable. Acyclic causal graphs have been of particular interest, implying an *asymmetry*: while changing the value of some variable $v$, we do not have to worry about dependencies that other variables might have on $v$. This knowledge has been exploited in a variety of ways among the planning community.

Among other things, acyclic causal graphs have been exploited to decompose planning instances into action hierarchies (Knoblock, 1994; Bacchus and Yang, 1994), to compute domain-independent heuristics for planning (Helmert, 2004), and to identify classes of planning instances that are easy to solve (Williams and Nayak, 1997; Jonsson and Bäckström, 1998; Braf-
man and Domshlak, 2003; Giménez and Jonsson, 2008; Jonsson, 2009). In each case, the resulting algorithm or procedure will not work if the causal graph is not acyclic. Thus one may be led to believe that planning is easier when the causal graph is acyclic. However, the exact complexity of planning over acyclic causal graphs has remained unknown: several researchers have shown that it is NP-hard (Domshlak and Dinitz, 2001; Brafman and Domshlak, 2003; Giménez and Jonsson, 2009), while planning is known to be PSPACE-complete in the general case of STRIPS (Bylander, 1994) and SAS+ (Bäckström and Nebel, 1995).

In this paper we close this complexity gap, establishing that the complexity of planning is PSPACE-complete for the class Acyc of planning instances with acyclic causal graphs. This result holds both for plan existence, the problem of determining whether there exists a solution to a given planning instance, and bounded plan existence, the problem of determining whether there exists a solution of bounded length. The results also holds for both STRIPS and SAS+ planning, although in the case of STRIPS our reduction requires the use of negative preconditions (when only positive preconditions are allowed, plan existence is known to be in P and bounded plan existence is NP-complete). As a consequence of our result, planning is no easier when the causal graph is acyclic, at least not in the worst case.

We also study two subclasses of Acyc: the class ISR-Acyc of planning instances with propositional variables that are either irreversible or symmetrically reversible, and the class SC-Acyc of planning instances with strongly connected domain transition graphs. We show that plan existence is NP-complete for ISR-Acyc and in P for SC-Acyc; i.e. in both cases, planning
becomes easier when we impose additional restrictions. We also show that bounded plan existence is \textbf{PSPACE}-hard for SC-Ayc, implying that optimal planning is significantly harder than satisficing planning for this class.

The work presented in this paper was previously published in the proceedings of the 2013 International Conference on Automated Planning and Scheduling (ICAPS). Compared to the conference publication, the present paper includes the following novel content:

- An encoding of the reduction from \texttt{Qbf-Sat} to plan existence for Acyc in PDDL, effectively translating quantified Boolean formulae to planning instances.

- A modification of the reduction such that actions have at most two preconditions, strengthening a previous result of Bylander (1994).

- An analysis showing that plan existence is \textbf{NP}-complete for the subclass ISR-Ayc of Acyc.

- A proof that bounded plan existence for SC-Ayc is \textbf{PSPACE}-complete, strengthening our previous result (which stated that it is \#\textbf{P}-hard).

The rest of the paper is organized as follows. Section 2 introduces the notation that we use in the paper. Section 3 describes how to define planning instances that simulate nested loops over an arbitrary number of propositional variables. Section 4 shows how to use these ideas to reduce \texttt{Qbf-Sat} to plan existence for Acyc, and describes a PDDL encoding of the reduction. Section 5 shows how to modify the previous reduction such that actions have at most two preconditions. Sections 6 and 7 study the complexity of the two
subclasses ISR-Ayc and SC-Ayc. Section 8 relates our results to previous work in the field, while Section 9 concludes with a discussion.

2. Notation

In this paper we study the complexity of both STRIPS and SAS$^+$ planning. To simplify the notation, we use a common description of planning instances that is valid for either formalism. The only difference between formalisms is the size of the variable domains, which equals two for STRIPS planning but is generally larger than two for SAS$^+$ planning. For STRIPS planning, since each variable $v$ is propositional, we use literals $v$ and $\overline{v}$ to describe the possible values of $v$ instead of an explicit domain such as $\{0, 1\}$.

Let $V$ be a set of variables, and let $D(v)$ be the finite domain of each variable $v \in V$. A partial state $p$ is a function on a subset of variables $V_p \subseteq V$ that maps each variable $v \in V_p$ to a value $p(v) \in D(v)$ in its domain. A state $s$ is a partial state such that $V_s = V$. The projection $p \mid U$ of a partial state $p$ onto a subset of variables $U \subseteq V$ is a partial state $q$ such that $V_q = V_p \cap U$ and $q(v) = p(v)$ for each $v \in V_q$. The composition $p \oplus q$ of two partial states $p$ and $q$ is a partial state $r$ such that $V_r = V_p \cup V_q$, $r(v) = q(v)$ for each $v \in V_q$, and $r(v) = p(v)$ for each $v \in V_p \setminus V_q$. Composition is not a commutative operation, but it is left associative.

When variables are propositional, we use sets of literals to define partial states, where each literal simultaneously defines a variable and its value. Given a subset $U \subseteq V$ of propositional variables, let $\overline{U} = \{ \overline{u} : u \in U \}$ denote the partial state with all variables in $U$ negated.

A planning instance is a tuple $P = \langle V, A, I, G \rangle$ where $V$ is a set of vari-
ables defined as above, $A$ is a set of actions with unit cost, $I$ is an initial state, and $G$ is a (partial) goal state. Each action $a = \langle \text{pre}(a), \text{post}(a) \rangle \in A$ has precondition $\text{pre}(a)$ and postcondition $\text{post}(a)$, both partial states on $V$. Action $a$ is applicable in state $s$ if $s \mid V_{\text{pre}(a)} = \text{pre}(a)$, and applying $a$ in $s$ results in a new state $s' = s \oplus \text{post}(a)$.

A plan is a sequence of actions $\langle a_1, \ldots, a_k \rangle$ such that $a_1$ is applicable in the initial state $I$ and, for each $2 \leq i \leq k$, $a_i$ is applicable in the state $I \oplus \text{post}(a_1) \oplus \cdots \oplus \text{post}(a_{i-1})$. The plan solves $P$ if the goal state is satisfied after applying $\langle a_1, \ldots, a_k \rangle$, i.e. if $(I \oplus \text{post}(a_1) \oplus \cdots \oplus \text{post}(a_k)) \mid V_G = G$. A landmark is a subgoal that must be achieved on every plan (in this paper we only consider subgoals on single variables).

The causal graph of $P$ is a directed graph $G = (V, E)$ with the variables of $P$ as nodes. There is an edge $(u, v) \in E$ if and only if there exists an action $a \in A$ such that $u \in V_{\text{pre}(a)} \cup V_{\text{post}(a)}$ and $v \in V_{\text{post}(a)}$. In this paper we focus on planning instances with acyclic causal graphs, implying that each action $a \in A$ is unary, i.e. satisfies $|V_{\text{post}(a)}| = 1$, since two or more variables in a postcondition would induce a cycle in the causal graph.

The domain transition graph (DTG) of a variable $v$ is a directed graph $\text{DTG}(v) = (D(v), E)$ with the values in the domain $D(v)$ of $v$ as nodes, and there is an edge $(x, y) \in E$ if and only if $x \neq y$ and there exists an action $a \in A$ such that $\text{post}(a)(v) = y$ and either $v \notin V_{\text{pre}(a)}$ or $\text{pre}(a)(v) = x$. $\text{DTG}(v)$ is strongly connected if and only if there is a directed path between $x$ and $y$ for each pair of values $x, y \in D(v)$.

A propositional variable $v \in V$ is irreversible if there exists no pair of actions $a, a' \in A$ such that $\text{post}(a) = \{v\}$ and $\text{post}(a') = \{\overline{v}\}$. Variable
$v \in V$ is \textit{symmetrically reversible} if for each action $a \in A$ such that $v \in V_{\text{post}}(a)$, there exists an action $a' \in A$ such that $\text{post}(a') = \text{post}(a)$ and $\text{pre}(a') \mid (V_{\text{pre}}(a) \setminus \{v\}) = \text{pre}(a) \mid (V_{\text{pre}}(a) \setminus \{v\})$, i.e. $a'$ and $a$ have opposite effects but the same precondition on variables other than $v$.

We define three classes of planning instances whose complexity we study in the paper:

- **Acyc**: planning instances with acyclic causal graphs.
- **ISR-Acyc**: the subclass of planning instances in Acyc with propositional variables that are either irreversible or symmetrically reversible.
- **SC-Acyc**: the subclass of planning instances in Acyc such that all variables have strongly connected DTGs.

Given an arbitrary planning instance $P$, we can check in polynomial time whether it belongs to Acyc, ISR-Acyc, and/or SC-Acyc.

For each class of planning instances $X$, we define $\text{PE}(X)$, the decision problem of plan existence for $X$, as follows:

**INPUT**: A planning instance $P \in X$.

**QUESTION**: Does there exist a plan solving $P$?

We also define the decision problem $\text{BPE}(X)$, the decision problem of bounded plan existence for $X$, as follows:

**INPUT**: A planning instance $P \in X$ and an integer $K$.

**QUESTION**: Is there a plan solving $P$ of length at most $K$?

Note that $\text{PE}(X)$ is polynomially reducible to $\text{BPE}(X)$ since each solvable planning instance must have a solution of length at most $K = \prod_{v \in V} |D(v)|$. 

7
Any longer plan must revisit states, and such a plan can always be shortened by removing all actions between a state and itself.

3. Loop Instances

In this section we introduce a novel mechanism for simulating nested loops using planning instances with propositional variables and acyclic causal graphs. There are examples in the literature of planning instances in Acyc that iterate over all assignments to \( n \) variables (Bäckström and Nebel, 1995), but none of these guarantee that assignments are not repeated, something that is crucial in our work. We describe our novel mechanism separately for two reasons: it is the most complicated part of our subsequent reductions, and it might have uses beyond those exploited in this paper.

3.1. Single Variable Loops

Our mechanism for simulating loops is based on a simple idea that we call loop instance, defined with respect to a specific planning instance.

**Definition 1.** Given a planning instance \( P = \langle V, A, I, G \rangle \), a loop instance is a subset \( U = \{a, b, x, u_1, u_2\} \subseteq V \) such that \( \{u_1, u_2\} \subseteq I \) and \( u_2 \) is a landmark of \( P \), and the only actions on \( U \) are those in the set \( A(U) \subseteq A \).

In other words, \( u_1 \) and \( u_2 \) are initially false, and either \( u_2 \in G \) or \( u_2 \) is a precondition of some action required to reach the goal state \( G \).

Figure 1 shows the actions in \( A(U) \), where \( v^1 \), \( v^2 \), etc. are the actions affecting a variable \( v \). The actions in \( A(U) \) may have preconditions other than those appearing in the table, which is why we refer to them as partial preconditions. The names of the variables in a loop instance may vary as
<table>
<thead>
<tr>
<th>Action</th>
<th>Partial precondition</th>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^1$</td>
<td>$\emptyset$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$b^1$</td>
<td>${\bar{a}}$</td>
<td>${b}$</td>
</tr>
<tr>
<td>$b^2$</td>
<td>${a}$</td>
<td>${\bar{b}}$</td>
</tr>
<tr>
<td>$x^1$</td>
<td>${a, b}$</td>
<td>${x}$</td>
</tr>
<tr>
<td>$x^2$</td>
<td>${b}$</td>
<td>${\bar{x}}$</td>
</tr>
<tr>
<td>$u^1_1$</td>
<td>${\bar{b}, x}$</td>
<td>${u_1}$</td>
</tr>
<tr>
<td>$u^1_2$</td>
<td>${\bar{b}, x, u_1}$</td>
<td>${u_2}$</td>
</tr>
</tbody>
</table>

Figure 1: The set of actions $A(U)$ and the associated causal graph of a loop instance $U$.

long as the associated actions match those in the set $A(U)$. Action $x^2$ is not strictly needed until later; its purpose in this section is to show that the subsequent lemmas hold even when it is present.

Figure 1 also shows the causal graph of a loop instance, which is in general a subgraph of the causal graph of its associated planning instance $P$. It is easy to verify that the causal graph is acyclic; a topological ordering is given by $a \rightarrow b \rightarrow x \rightarrow u_1 \rightarrow u_2$.

We proceed to prove several lemmas regarding loop instances. Although the initial state only explicitly mentions variables $u_1$ and $u_2$, the remaining
variables have to be initially false for a loop instance to be solvable. Moreover, the solution always contains a unique subsequence of actions.

**Lemma 2.** Given a planning instance \( P \) with associated loop instance \( U = \{a, b, x, u_1, u_2\} \), the following holds:

1. \( P \) is unsolvable unless the variables in \( \{a, b, x\} \) are initially false.
2. Any plan solving \( P \) contains the subsequence \( \langle u_1^1, b^1, a^1, x^1, b^2, u_2^1 \rangle \).

*Proof.* We first assume that the variables in \( \{a, b, x\} \) are initially false and prove the second part of the lemma. Since \( u_2 \) is a landmark of \( P \), any plan solving \( P \) has to apply action \( u_2^1 \) with precondition \( u_1 \), requiring action \( u_1^1 \) to appear before \( u_2^1 \). The precondition \( \{ \overline{b}, \overline{x} \} \) of \( u_1^1 \) is satisfied in the initial state, but to satisfy the precondition \( x \) of \( u_2^1 \), action \( x^1 \) has to appear between \( u_1^1 \) and \( u_2^1 \). The precondition \( b \) of \( x^1 \) requires \( b^1 \) to appear between \( u_1^1 \) and \( x^1 \), and the precondition \( \overline{b} \) of \( u_2^1 \) requires \( b^2 \) between \( x^1 \) and \( u_2^1 \). Finally, the precondition \( a \) of \( x^1 \) and \( b^2 \) requires \( a^1 \) between \( b^1 \) and \( x^1 \). Taken together, this results in the unique subsequence \( \langle u_1^1, b^1, a^1, x^1, b^2, u_2^1 \rangle \).

We next show that the variables in \( \{a, b, x\} \) have to be false before applying action \( u_1^1 \) for \( P \) to be solvable. If \( a \) is initially true, there is no action making \( a \) false, rendering it impossible to satisfy the precondition \( \overline{x} \) of \( b^1 \) required to make \( b \) true between \( u_1^1 \) and \( x^1 \). If \( b \) is initially true, we have to apply action \( b^2 \) to satisfy the precondition \( \{ \overline{b}, \overline{x} \} \) of \( u_1^1 \), which in turn requires \( a \) to be true. Finally, if \( x \) is initially true we have to apply action \( x^2 \) to satisfy the precondition \( \{ \overline{b}, \overline{x} \} \) of \( u_1^1 \), which in turn requires \( b \) to be true. \( \square \)

We also show that the values of the variables in the subset \( \{a, b, x\} \) are fixed before action \( u_1^1 \) and after action \( u_2^1 \).
Lemma 3. No plan solving $P$ can change the value of a variable in the subset \{a, b, x\} before action $u_1$ or after action $u_2$.

Proof. In the proof of Lemma 2 we already showed that the variables in \{a, b, x\} have to be false before applying action $u_1$, implying that no action can change the value of a variable in \{a, b, x\} before $u_1$. The action subsequence from Lemma 2 applies action $u_2$ in the partial state \{a, b, x\}, in which no action on \{a, b, x\} is applicable, making it impossible for any plan to change the value of a variable in \{a, b, x\} after $u_2$.

The name “loop instance” derives from the fact that variable $x$ is false before action $u_1$ and true after $u_2$, causing any solution to “iterate” over the two possible values of $x$. Another direct consequence of Lemma 2 is that the variables in \{a, b, x\} have to be false in the initial state; we can therefore say that a loop instance $U$ has implicit initial state $\overline{U}$.

3.2. Nested loops on two variables

In this section we show how to combine loop instances to represent nested loops on two variables. Consider a planning instance $P_2 = \langle V, A, I, G \rangle$ with the following components:

- $V = \{a_1, b_1, x_1, u_{10}, u_{11}, u_{12}, u_{13}\} \cup \{a_2, b_2, x_2, u_{21}, u_{22}\}$,
- $I = \overline{V}$,
- $G = \{u_{13}\}$.

Table 1 shows the actions of the planning instance $P_2$. It is easy to verify that $U_1 = \{a_1, b_1, x_1, u_{11}, u_{12}\}$ is a loop instance: the actions on these variables match those in Figure 1, $u_{11}$ and $u_{12}$ are initially false, and $u_{12}$ is a
<table>
<thead>
<tr>
<th>Action</th>
<th>Precondition</th>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1^1$</td>
<td>$\emptyset$</td>
<td>${a_1}$</td>
</tr>
<tr>
<td>$b_1^1$</td>
<td>${\overline{a}_1}$</td>
<td>${b_1}$</td>
</tr>
<tr>
<td>$b_1^2$</td>
<td>${a_1}$</td>
<td>$\overline{b}_1$</td>
</tr>
<tr>
<td>$x_1^1$</td>
<td>${a_1, b_1}$</td>
<td>${x_1}$</td>
</tr>
<tr>
<td>$x_1^2$</td>
<td>${b_1}$</td>
<td>$\overline{x}_1$</td>
</tr>
<tr>
<td>$a_2^1$</td>
<td>$\emptyset$</td>
<td>${a_2}$</td>
</tr>
<tr>
<td>$b_2^1$</td>
<td>${\overline{a}_2}$</td>
<td>${b_2}$</td>
</tr>
<tr>
<td>$b_2^2$</td>
<td>${a_2}$</td>
<td>$\overline{b}_2$</td>
</tr>
<tr>
<td>$x_2^1$</td>
<td>${a_2, b_2}$</td>
<td>${x_2}$</td>
</tr>
<tr>
<td>$x_2^2$</td>
<td>${b_2}$</td>
<td>$\overline{x}_2$</td>
</tr>
<tr>
<td>$u_{10}^1$</td>
<td>${\overline{u}<em>{21}, \overline{u}</em>{22}}$</td>
<td>${u_{10}}$</td>
</tr>
<tr>
<td>$u_{11}^1$</td>
<td>$\overline{b}<em>1, x_1 \cup {u</em>{10}, u_{22}}$</td>
<td>${u_{11}}$</td>
</tr>
<tr>
<td>$u_{12}^1$</td>
<td>$\overline{b}<em>1, x_1, u</em>{11} \cup {\overline{u}<em>{21}, \overline{u}</em>{22}}$</td>
<td>${u_{12}}$</td>
</tr>
<tr>
<td>$u_{13}^1$</td>
<td>${u_{12}, u_{22}}$</td>
<td>${u_{13}}$</td>
</tr>
<tr>
<td>$u_{21}^1$</td>
<td>$\overline{b}_2, \overline{x}_2$</td>
<td>${u_{21}}$</td>
</tr>
<tr>
<td>$u_{22}^1$</td>
<td>$\overline{b}<em>2, x_2, u</em>{21}$</td>
<td>${u_{22}}$</td>
</tr>
<tr>
<td>$a_2^2$</td>
<td>${b_1}$</td>
<td>$\overline{a}_2$</td>
</tr>
<tr>
<td>$u_{21}^2$</td>
<td>${b_1}$</td>
<td>$\overline{u}_{21}$</td>
</tr>
<tr>
<td>$u_{22}^2$</td>
<td>${b_1}$</td>
<td>$\overline{u}_{22}$</td>
</tr>
</tbody>
</table>

Table 1: The actions of the planning instance $P_2$. 

precondition of the only action $u_{13}^1$ that adds the goal $u_{13}$. For clarity, we have separated the partial preconditions of actions $u_{11}^1$ and $u_{12}^1$ that match those in Figure 1.
The subset $U_2 = \{a_2, b_2, x_2, u_{21}, u_{22}\}$ is also similar to a loop instance, but if we compare to Figure 1, $P_2$ has three additional actions $a_2^2$, $u_{21}^2$, and $u_{22}^2$, each with precondition $b_1$ and making the corresponding variable false. We refer to $U_2$ as a conditional loop instance: whenever $b_1$ is false, the actions on $U_2$ are exactly those of a loop instance, but when $b_1$ is true, the properties of a loop instance no longer hold.

**Definition 4.** Given a planning instance $P = \langle V, A, I, G \rangle$, a conditional loop instance $U = \{a, b, x, u_1, u_2\}$ is a loop instance with three additional actions $a_2 = \langle \{v\}, \{\overline{v}\} \rangle$, $u_1^2 = \langle \{v\}, \{\overline{u_1}\} \rangle$, and $u_2^2 = \langle \{v\}, \{\overline{u_2}\} \rangle$, where $v \notin U$. We say that $U$ is conditional on $v$ and activated whenever $v$ is false.

A secondary function of loop instances is to activate and deactivate other, conditional, loop instances. For example, the loop instance $U_1$ of $P_2$ regulates the conditional loop instance $U_2$ by means of the variable $b_1$. Lemma 3 implies that $b_1$ is false before action $u_{11}^1$ and after action $u_{12}^1$; as a consequence, conditional loop instance $U_2$ is activated at those portions of the plan.

Let us study the structure of a plan for $P_2$. Due to Lemma 2 and the fact that $U_1$ is a loop instance, any plan for $P_2$ contains the subsequence $\langle u_{11}^1, b_1^1, a_1^1, x_1^1, b_1^2, u_{12}^1 \rangle$. The precondition $u_{10}^1$ of $u_{11}^1$ requires action $u_{10}^1$ to appear before $u_{11}^1$, and action $u_{13}^1$, needed to achieve the goal $u_{13}$, has to appear after $u_{12}^1$ because of its precondition $u_{12}$.

As a consequence, any plan solving $P_2$ contains the action subsequence $\langle u_{10}^1, u_{11}^1, b_1^1, a_1^1, x_1^1, b_1^2, u_{12}^1, u_{13}^1 \rangle$. Actions $u_{10}^1$ and $u_{12}^1$ both have precondition $\{\overline{u_{21}}, \overline{u_{22}}\}$, and $u_{11}^1$ and $u_{13}^1$ have precondition $\{u_{22}\}$. As previously mentioned, Lemma 3 implies that $b_1$ is false before $u_{11}^1$ and after $u_{12}^1$, causing the conditional loop instance $U_2$ to be activated.
To apply the two pairs of actions \((u_{10}^1, u_{11}^1)\) and \((u_{12}^1, u_{13}^1)\), we have to make \(u_{22}^1\) true starting from \(\{\overline{u}_{21}, u_{22}^1\}\) while \(U_2\) is activated. This corresponds to a partial planning instance \(P'_2 = \langle U_2, A(U_2), \overline{U}_2, \{u_{22}^1\}\rangle\) with associated loop instance \(U_2\), where \(\overline{U}_2\) is the implicit initial state required for \(P'_2\) to be solvable. Lemma 2 implies that any plan solving \(P'_2\) contains the subsequence \(\omega_2 = \langle u_{21}^1, b_1^2, a_2^1, x_2^1, b_2^2, u_{22}^1\rangle\). In a plan solving the original instance \(P_2\), \(\omega_2\) has to be appended between \(u_{10}^1\) and \(u_{11}^1\), and between \(u_{12}^1\) and \(u_{13}^1\).

When we apply action \(u_{11}^1\), the partial state on \(U_2\) is \(\{a_2^1, \overline{b}_2, x_2, u_{21}, u_{22}\}\). Prior to applying action \(u_{12}^1\), we have to reset the variables in \(U_2\) to false to achieve the implicit initial state \(\overline{U}_2\) of \(P'_2\). When \(b_1^1\) is true, the action sequence \(\rho_2 = \langle a_2^2, b_1^1, x_2^2, a_2^1, b_2^2, a_2^1, u_{21}^1, u_{22}^1\rangle\) first resets the variables in \(\{a_2, b_2, x_2\}\) to false, and then variables \(u_{21}\) and \(u_{22}\).

Summarizing, a plan for \(P_2\) is of the form

\[
\langle u_{10}^1, \omega_2, u_{11}^1, b_1^1, \rho_2, a_1^1, x_1^1, b_1^2, u_{12}^1, \omega_2, u_{13}^1\rangle.
\]

Some actions can change places, but \(\omega_2\) has to appear between \(u_{10}^1\) and \(u_{11}^1\) and between \(u_{12}^1\) and \(u_{13}^1\). Likewise, \(\rho_2\) has to appear between \(b_1^1\) and \(b_1^2\) (where \(b_1^1\) is true). A plan for \(P_2\) describes a nested loop on \(x_1\) and \(x_2\): variable \(x_1\) is false before \(u_{11}^1\) and true after \(u_{12}^1\) and, for each of the two values on \(x_1\), variable \(x_2\) is false before \(u_{21}^1\) and true after \(u_{22}^1\) in the subsequence \(\omega_2\).

Note that if variables \(u_{10}, \ldots, u_{13}\) are initially false, all other variables in \(V\) have to be false for \(P_2\) to be solvable. Since \(U_1\) is a loop instance, the variables in \(\{a_1, b_1, x_1\}\) have to be initially false due to Lemma 2. Since \(b_1^1\) is false before action \(u_{11}^1\), conditional loop instance \(U_2\) is activated, and to apply the action pair \((u_{10}^1, u_{11}^1)\) we have to solve the partial planning instance \(P'_2\) with implicit initial state \(\overline{U}_2\).
Figure 2: The causal graph of the planning instance $P_2$.

Figure 2 shows the causal graph of the planning instance $P_2$. The subgraphs on $U_1$ and $U_2$ are those of a loop instance. There are edges from $b_1$ to $a_2$, $u_{21}$, and $u_{22}$ due to $U_2$ being a loop instance conditional on $b_1$, and edges from $u_{21}$ and $u_{22}$ to $u_{10}, \ldots, u_{13}$ due to actions $u_{10}^1, \ldots, u_{13}^1$. By stacking $a_1, b_1, x_1$ at the top and $u_{10}, \ldots, u_{13}$ at the bottom, a topological ordering is given by scanning variables left-to-right in each row, starting from the top.

### 3.3. Nested loops on $n$ variables

We next show how to extend the idea from the previous section to simulate nested loops on any number $n$ of variables. Consider a planning instance $P_n = \langle V, A, I, G \rangle$ with the following components:

- $V = \bigcup_{i=1}^{n} (X_i \cup V_i)$,

- $X_i = \{a_i, b_i, x_i\}$ for each $1 \leq i \leq n$. 


\[ V_i = \{u_{i0}, u_{i1}, u_{i2}, u_{i3}\} \text{ for each } 1 \leq i \leq n, \]

\[ I = \emptyset, \]

\[ G = \{u_{i3}\}. \]

Table 2 shows the actions of the planning instance \( P_n \) for \( 1 < i \leq n \) and \( 0 \leq k \leq 3 \). For each \( 1 \leq i \leq n \), let \( U_i = \{a_i, b_i, x_i, u_{i1}, u_{i2}\} \), let \( A_i \subseteq A \) be the subset of actions on variables in \( X_i \), and let \( B_i \subseteq A \) be the actions on \( V_i \). The actions are defined such that \( U_1 \) is a loop instance and, for each \( 1 < i \leq n \), \( U_i \) is a loop instance conditional on \( b_{i-1} \).

Compared to the planning instance \( P_2 \) from the previous section, the set \( V_i, 1 < i \leq n \), contains additional variables \( u_{i0} \) and \( u_{i3} \) with associated actions in \( B_i \). Consequently, actions \( u_{(i-1)0}^1 \) and \( u_{(i-1)2}^1 \) have partial precondition \( \{u_{i0}, u_{i1}, u_{i2}, u_{i3}\} \), and actions \( u_{(i-1)1}^1 \) and \( u_{(i-1)3}^1 \) have partial precondition \( \{u_{i3}\} \). These extra variables are not strictly needed for \( i = n \) but including them makes action definitions uniform (and hence more compact).

For each \( 1 \leq i \leq n \), define a partial planning instance \( P'_i = \langle V'_i, A'_i, I'_i, G'_i \rangle \) with components

\[ V'_i = \bigcup_{j=i}^n (X_j \cup V_j), \]

\[ A'_i = \bigcup_{j=i}^n (A_j \cup B_j) \setminus \bigcup_{j=2}^i \{a_j^2, u_{j1}, u_{j2}\}, \]

\[ I'_i = \overline{V}'_i, \]

\[ G'_i = \{u_{i3}\}. \]

**Lemma 5.** For each \( 1 \leq i \leq n \), any solution to \( P'_i \) iterates over all possible assignments to variables \( x_i, \ldots, x_n \).
<table>
<thead>
<tr>
<th>Action</th>
<th>Precondition</th>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1^1$</td>
<td>$\emptyset$</td>
<td>${a_1}$</td>
</tr>
<tr>
<td>$b_1^1$</td>
<td>${\overline{a}_1}$</td>
<td>${b_1}$</td>
</tr>
<tr>
<td>$b_1^2$</td>
<td>${a_1}$</td>
<td>${\overline{b}_1}$</td>
</tr>
<tr>
<td>$x_1^1$</td>
<td>${a_1, b_1}$</td>
<td>${x_1}$</td>
</tr>
<tr>
<td>$x_1^2$</td>
<td>${b_1}$</td>
<td>${\overline{x}_1}$</td>
</tr>
<tr>
<td>$a_i^1$</td>
<td>$\emptyset$</td>
<td>${a_i}$</td>
</tr>
<tr>
<td>$b_i^1$</td>
<td>${\overline{a}_i}$</td>
<td>${b_i}$</td>
</tr>
<tr>
<td>$b_i^2$</td>
<td>${a_i}$</td>
<td>${\overline{b}_i}$</td>
</tr>
<tr>
<td>$x_i^1$</td>
<td>${a_i, b_i}$</td>
<td>${x_i}$</td>
</tr>
<tr>
<td>$x_i^2$</td>
<td>${b_i}$</td>
<td>${\overline{x}_i}$</td>
</tr>
<tr>
<td>$u_{i-1,0}^1$</td>
<td>${\overline{u}<em>0, \overline{u}</em>{i1}, \overline{u}<em>{i2}, \overline{u}</em>{i3}}$</td>
<td>${u_{(i-1)0}}$</td>
</tr>
<tr>
<td>$u_{i-1,1}^1$</td>
<td>${\overline{b}<em>{i-1}, \overline{x}</em>{i-1}} \cup {u_{(i-1)0}, u_{i3}}$</td>
<td>${u_{(i-1)1}}$</td>
</tr>
<tr>
<td>$u_{i-1,2}^1$</td>
<td>${\overline{b}<em>{i-1}, x</em>{i-1}, u_{(i-1)1}} \cup {\overline{u}<em>0, \overline{u}</em>{i1}, \overline{u}<em>{i2}, \overline{u}</em>{i3}}$</td>
<td>${u_{(i-1)2}}$</td>
</tr>
<tr>
<td>$u_{i-1,3}^1$</td>
<td>${u_{(i-1)2}, u_{i3}}$</td>
<td>${u_{(i-1)3}}$</td>
</tr>
<tr>
<td>$u_{n0}^1$</td>
<td>$\emptyset$</td>
<td>${u_{n0}}$</td>
</tr>
<tr>
<td>$u_{n1}^1$</td>
<td>${\overline{b}_n, \overline{x}<em>n} \cup {u</em>{n0}}$</td>
<td>${u_{n1}}$</td>
</tr>
<tr>
<td>$u_{n2}^1$</td>
<td>${\overline{b}<em>n, x_n, u</em>{n1}}$</td>
<td>${u_{n2}}$</td>
</tr>
<tr>
<td>$u_{n3}^1$</td>
<td>${u_{n2}}$</td>
<td>${u_{n3}}$</td>
</tr>
<tr>
<td>$a_i^2$</td>
<td>${b_{i-1}}$</td>
<td>${\overline{a}_i}$</td>
</tr>
<tr>
<td>$u_{ik}^2$</td>
<td>${b_{i-1}}$</td>
<td>${\overline{a}_k}$</td>
</tr>
</tbody>
</table>

Table 2: The actions of the planning instance $P_n$ for $1 < i \leq n$ and $0 \leq k \leq 3$.

**Proof.** With $\{a_i^2, u_{i1}^2, u_{i2}^2\}$ removed, the actions of $U_i$ are exactly those of a loop instance. Variable $u_{i2}$ is a precondition of the only action $u_{i3}^1$ that
achieves the goal \( u_{i3} \), and \( u_{i1} \) and \( u_{i2} \) are initially false. Lemma 2 states that any plan solving \( P'_i \) contains the action subsequence \( \langle u_{i1}^1, b_{i1}^1, a_{i1}^1, x_{i1}^1, b_{i2}^2, u_{i2}^1 \rangle \).

We now prove the lemma by induction on \( i \). For \( i = n \), since \( U_n \) is a loop instance for \( P'_n \), any plan solving \( P'_n \) iterates over the two possible values of \( x_n \). For \( i < n \), because of the way actions are defined, \( u_{i0}^1 \) has to appear before \( u_{i1}^1 \), and \( u_{i3}^1 \) has to appear after \( u_{i2}^1 \). The action pairs \( (u_{i0}^1, u_{i1}^1) \) and \( (u_{i2}^1, u_{i3}^1) \) each requires achieving \( u_{(i+1)3} \) starting from \( \{\overline{u}_{(i+1)0}, \overline{u}_{(i+1)1}, \overline{u}_{(i+1)2}, \overline{u}_{(i+1)3}\} \).

Due to Lemma 3, variable \( b_i \) is false while doing so, causing conditional loop instance \( U_{i+1} \) to be activated. This corresponds to solving the partial planning problem \( P'_{i+1} \), which has implicit initial state \( V'_{i+1} \) since variables in \( \{a_{i+1}, b_{i+1}, x_{i+1}\} \) have to be false for \( P'_{i+1} \) to be solvable due to Lemma 2 and, if \( i + 1 < n \), we recursively have to satisfy the implicit initial state \( V'_{i+2} \) of \( P'_{i+2} \) to apply the action pair \( (u_{(i+1)0}^1, u_{(i+1)1}^1) \) while \( b_{i+1} \) is false.

Let \( \omega_{i+1} \) be a plan solving \( P'_{i+1} \). By hypothesis of induction, \( \omega_{i+1} \) iterates over all possible values to variables \( x_{i+1}, \ldots, x_n \). A plan for \( P'_i \) is given by

\[
\langle u_{i0}^1, \omega_{i+1}, u_{i1}^1, b_{i1}^1, \rho_{i+1}, a_{i1}^1, x_{i1}^1, b_{i2}^2, u_{i2}^1, \omega_{i+1}, u_{i3}^1 \rangle,
\]

where \( \rho_{i+1} \) is an action sequence resetting the variables in \( V'_{i+1} \) to false. Since \( \omega_{i+1} \) appears before \( u_{i1}^1 \) and after \( u_{i2}^1 \), this plan iterates over all possible assignments to variables \( x_{i+1}, \ldots, x_n \), first for \( x_i \) false, then for \( x_i \) true. This corresponds exactly to iterating over all assignments to \( x_i, \ldots, x_n \). 

We omit the causal graph of \( P_n \), which has the same structure as the causal graph of \( P_2 \): a topological ordering is given by

\[
a_1 \rightarrow b_1 \rightarrow x_1 \rightarrow \cdots \rightarrow a_n \rightarrow b_n \rightarrow x_n \rightarrow \]

\[
\rightarrow u_{n0} \rightarrow u_{n1} \rightarrow u_{n2} \rightarrow u_{n3} \rightarrow \cdots \rightarrow u_{10} \rightarrow u_{11} \rightarrow u_{12} \rightarrow u_{13}.
\]
It is possible to verify that no action on a variable $v$ has a precondition on a variable $u$ appearing after $v$ in this ordering.

3.4. Case Study: Reducing Unsat to Pe(Ayc)

In this section, we show how to use loop instances to reduce the decision problem Unsat to Pe(Ayc). Prior to our work, it was not known how to do this, not even for SAS$^+$ planning. Using loop instances, the reduction is almost trivial.

Let $\phi = (c_1 \land \cdots \land c_m)$ be a 3SAT formula on $n$ variables $x_1, \ldots, x_n$ where, for each $1 \leq j \leq m$, $c_j = \ell^1_j \lor \ell^2_j \lor \ell^3_j$ is a 3-literal clause on $x_1, \ldots, x_n$. The decision problem Unsat consists in determining whether or not $\phi$ is unsatisfied for all assignments to $x_1, \ldots, x_n$.

Given $\phi$, we construct (in polynomial time) a planning instance in Acyc by modifying the planning instance $P_n$ from the previous section. The only modification is replacing actions $u^1_{n1}$ and $u^1_{n2}$ with $m$ actions each, corresponding to the clauses $c_1, \ldots, c_m$ of $\phi$. For each $1 \leq j \leq m$, the actions $u^j_{n1}$ and $u^j_{n2}$ associated with $c_j$ have additional precondition $\{\ell^1_j, \ell^2_j, \ell^3_j\}$ where, for each $1 \leq k \leq 3$, $\ell^k_j = x_i$ if $\ell^k_j = x_i$ for some variable $x_i$, and $\ell^k_j = \bar{x}_i$. Technically, a loop instance on $U_n$ should only have one action $u^1_{n1}$ and one action $u^1_{n2}$, but $U_n$ still shares all the properties of a loop instance since each of the new actions has the same precondition as the original $u^1_{n1}$ or $u^1_{n2}$.

**Lemma 6.** The modified planning instance $P_n$ has a solution if and only if $\phi$ is unsatisfiable.

**Proof.** Lemma 5 states that a plan solving $P_n$ iterates over all possible assignments to variables $x_1, \ldots, x_n$. Since the innermost loop is over $x_n$, for
each such assignment we have to apply one of the actions $u_{n1}^j$ or $u_{n2}^j$. If $\phi$ is unsatisfied, for each assignment to $x_1, \ldots, x_n$ there exists an unsatisfied clause $c_j$, making $u_{n1}^j$ or $u_{n2}^j$ applicable. On the other hand, if there exists an assignment to $x_1, \ldots, x_n$ that satisfies $\phi$, none of the actions $u_{n1}^j$ or $u_{n2}^j$ are applicable, breaking the chain and rendering $P_n$ unsolvable.

4. The Complexity of Planning for Acyc

In this section we show that the decision problem $PE(Ayc)$ is $PSPACE$-complete by reduction from $QBF$-$Sat$. The reduction makes heavy use of the loop instances introduced in the previous section. We first introduce the decision problem $QBF$-$Sat$ and describe a general strategy for solving it. We then show how to construct a planning instance in Acyc that simulates this strategy, and finally prove that the reduction is correct.

4.1. The Decision Problem $QBF$-$Sat$

A quantified Boolean formula, or QBF, is a conjunction of clauses such that the variables are bound by quantifiers, either existential or universal. The decision problem $QBF$-$Sat$ is to determine whether a given QBF $F$ is satisfiable, and is known to be $PSPACE$-complete (Stockmeyer and Meyer, 1973), even when $F$ is in prenex normal form, i.e. the quantifiers alternate between existential and universal. Although the reduction from $QBF$-$Sat$ to $PE(Ayc)$ can be implemented for any QBF, the resulting planning instance in Acyc is significantly simpler when the QBF is in prenex normal form.

A QBF in prenex normal form is a formula $F = \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n \cdot \phi$, where $n$ is an even integer, $\phi = (c_1 \land \cdots \land c_m)$ is a 3SAT formula, and
1 function QSat(i, p_i)
2     if i = n then
3         return Check(p_i ∪ {\overline{x_i}}) or Check(p_i ∪ {x_i})
4     else if i is odd then
5         return QSat(i + 1, p_i ∪ {\overline{x_i}}) and QSat(i + 1, p_i ∪ {x_i})
6     else
7         return QSat(i + 1, p_i ∪ {\overline{x_i}}) or QSat(i + 1, p_i ∪ {x_i})

Figure 3: Algorithm QSat that checks if \( F_i(p_i) \) is satisfiable.

c_j = \ell^1_j \lor \ell^2_j \lor \ell^3_j \) is a 3-literal clause for each 1 ≤ j ≤ m. In what follows we describe a general algorithm for determining whether F is satisfiable.

For each 1 ≤ i ≤ n, let p_i be a partial state representing an assignment to the variables x_1, ..., x_{i-1}. Let \( F_i(p_i) = Q_i x_i \cdots \forall x_{n-1} \exists x_n \cdot \phi(p_i) \) denote the partial QBF obtained from F by removing the quantifiers on x_1, ..., x_{i-1} and replacing x_1, ..., x_{i-1} in \( \phi \) with the respective truth values in p_i. Figure 3 describes a recursive algorithm QSat that checks whether \( F_i(p_i) \) is satisfiable for any arbitrary 1 ≤ i ≤ n and p_i. The algorithm Check(p_{n+1}) returns true if and only if the 3SAT formula \( \phi \) is satisfied by the assignment p_{n+1}. Note that \( F_1(p_1) = F_1(\emptyset) = F \), so the following lemma implies that F is satisfiable if and only if QSat(1, \emptyset) returns true.

Lemma 7. The algorithm QSat runs in polynomial space on input (i, p_i) and returns true if and only if \( F_i(p_i) \) is satisfiable.

Proof. The recursive algorithm QSat essentially performs a nested loop on the variables x_i, ..., x_n with the body in the inner loop described by a call
to Check. The proof follows directly from the meaning of each quantifier. If \( i = n \), the quantifier on \( x_i \) is existential, and \( F_n(p_n) \) is satisfiable if and only if \( \phi \) is satisfiable for either of the assignments \( p_n \cup \{ \overline{x}_n \} \) or \( p_n \cup \{ x_n \} \). If \( i \) is odd, \( x_i \) is universal, so \( F_i(p_i) \) is satisfiable if and only if \( F_{i+1}(p_i \cup \{ \overline{x}_i \}) \) and \( F_{i+1}(p_i \cup \{ x_i \}) \) are satisfiable. Otherwise \( x_i \) is existential, so \( F_i(p_i) \) is satisfiable if and only if \( F_{i+1}(p_i \cup \{ \overline{x}_i \}) \) or \( F_{i+1}(p_i \cup \{ x_i \}) \) is satisfiable.

By sharing the memory needed to store \( p_{n+1} \) (which requires \( O(n) \) space), each recursive call only needs \( O(\log i) = O(\log n) \) memory to represent \( i \), and a single bit of memory to remember the outcome of Check or QSAT for \( p_i \cup \overline{x}_i \). Checking whether an assignment \( p_{n+1} \) satisfies \( \phi \) requires \( O(n + m) \) space where \( m \) is the number of clauses, and the recursive calls require a total of \( O(n \log n) \) space since there are never more than \( n \) such calls on the stack. Thus QSAT runs in \( O(n \log n + m) \) space, which is polynomial in \( F \). □

4.2. Construction

In this section we show how to reduce the decision problem QBF-SAT to PE(AcyC). Specifically, for any QBF \( F \) in prenex normal form, we construct a planning instance in Acyc that is solvable if and only if \( F \) is satisfiable.

Let \( F = \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n \cdot \phi \) be the QBF in prenex normal form with \( \phi = (c_1 \land \cdots \land c_m) \) and \( c_j = \ell_{j,1}^1 \lor \ell_{j,2}^2 \lor \ell_{j,3}^3 \) for each \( 1 \leq j \leq m \). Given \( F \), we construct a planning instance \( P_F = \langle \mathcal{V}_F, \mathcal{A}_F, \mathcal{I}_F, \mathcal{G}_F \rangle \) where

- \( \mathcal{V}_F = \bigcup_{i=1}^n (X_i \cup V_i) \cup S \),
- \( X_i = \{ a_i, b_i, x_i \} \) for each \( 1 \leq i \leq n \),
- \( V_i = \{ u_{i0}, u_{i1}, u_{i2}, u_{i3}, v_{i1}, v_{i2}, v_{i3} \} \) for each \( 1 \leq i \leq n \),

22
Table 3: Actions in the set $A_S$ for $1 \leq j \leq m$. The precondition $s_{j-1}$ is omitted for $j = 1$.

- $S = \{s_1, \ldots, s_m, t\}$,
- $A_F = \bigcup_{i=1}^n (A_i \cup B_i) \cup A_S$,
- $I_F = \nabla F$,
- $G_F = \{u_{13}\}$.

For each $1 \leq i \leq n$, the set of actions $A_i$ on $X_i = \{a_i, b_i, x_i\}$ are exactly those of the planning instance $P_n$ from the previous section.

However, if we compare to $P_n$, $V_i$ contains additional variables $v_{i1}, v_{i2}, v_{i3}$. To implement the algorithm QSat in Figure 3, we need to remember whether or not the partial QBF $F_i(p_i)$ is satisfiable. The variables in $V_i$ constitute a simple memory for doing so. As a consequence, we cannot immediately apply the results regarding loop instances from the previous section.

The purpose of the variables in the set $S$ is to implement the subroutine Check, i.e. to verify whether the 3SAT formula $\phi$ is satisfied given the current assignment $p_{n+1}$ to the variables $x_1, \ldots, x_n$. Table 3 shows the actions in the
associated set $A_S$, with the precondition $s_{j-1}$ of actions $s^1_j$, $s^2_j$, and $s^3_j$ omitted for $j = 1$. For each $1 \leq j \leq m$, literals $\ell^1_j$, $\ell^2_j$, and $\ell^3_j$ should be replaced with the corresponding variable among $x_1, \ldots, x_n$, appropriately negated.

The actions in $A_S$ are defined such that starting from $S$, we can make $s_m$ true if and only if $\phi$ is satisfied, and $t$ true if and only if $\phi$ is unsatisfied. To make $t$ true, it is sufficient to find a clause $c_j$ that is unsatisfied by the current assignment to $x_1, \ldots, x_n$, and apply the associated action $t^j$. To make $s_m$ true, we have to verify that each clause is satisfied by the current assignment to $x_1, \ldots, x_n$. This is the reason why actions $s^1_j$, $s^2_j$, and $s^3_j$ have precondition $s_{j-1}$. The purpose of actions $s^4_j$, $1 \leq j \leq m$, and $t^{m+1}$ is to reset the variables in $S$ to false when $b_n$ is true.

Table 4 shows the actions in the set $B_1 \cup \cdots \cup B_n$ for $1 \leq i < n$ and $0 \leq k \leq 3$. For each $1 \leq i \leq n$, the two subsets $U_{i1} = \{a_i, b_i, x_i, u_{i1}, u_{i2}\}$ and $U_{i2} = \{a_i, b_i, x_i, v_{i1}, v_{i2}\}$ are similar to loop instances, conditional on $b_{i-1}$ for $i > 1$, but $u_{i2}$ and $v_{i2}$ are not always landmarks, violating the definition of loop instances. In these cases, $\{u_{i2}, v_{i2}\}$ is a disjunctive landmark, implying that any plan has to use one of $U_{i1}$ and $U_{i2}$ to achieve the goal.

For $1 \leq i \leq n$, let $p_i$ be an assignment to $x_1, \ldots, x_{i-1}$. We define a partial planning instance $P'_{i1}(p_i) = \langle V'_i, A'_i, I'_i, \{u_3\} \rangle$ as follows:

- $V'_i = Z'_i \cup \{x_1, \ldots, x_{i-1}\}$,
- $Z'_i = \bigcup_{j=i}^n (X_j \cup V_j) \cup S$,
- $A'_i = \bigcup_{j=i}^n (A_j \cup B_j) \cup A_S \setminus \bigcup_{j=2}^i \{a^2_j, u^4_j, v^4_j\}$,
- $I'_i = \overline{Z_i'} \cup p_i$.
<table>
<thead>
<tr>
<th>Action</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{i0}$</td>
<td>$\neg v_{i+1}$</td>
<td>${u_{i0}}$</td>
</tr>
<tr>
<td>$u_{i1}$</td>
<td>${\overline{b}<em>i, x_i} \cup {u</em>{i0}, v_{(i+1)3}}$</td>
<td>${u_{i1}}$</td>
</tr>
<tr>
<td>$u_{i2}$</td>
<td>${\overline{b}<em>i, x_i, u</em>{i1}} \cup \neg v_{i+1}$</td>
<td>${u_{i2}}$</td>
</tr>
<tr>
<td>$u_{i3}$</td>
<td>${u_{i2}, v_{(i+1)3}}$</td>
<td>${u_{i3}}$</td>
</tr>
<tr>
<td>$v_{i1}$</td>
<td>${\overline{b}<em>i, x_i} \cup {u</em>{i0}, u_{(i+1)3}}$</td>
<td>${v_{i1}}$</td>
</tr>
<tr>
<td>$v_{i2}$</td>
<td>${\overline{b}<em>i, x_i, v</em>{i1}} \cup \neg v_{i+1}$</td>
<td>${v_{i2}}$</td>
</tr>
<tr>
<td>$v_{i3}$</td>
<td>${u_{i2}, u_{(i+1)3}}$</td>
<td>${v_{i3}}$</td>
</tr>
<tr>
<td>$v_{i3}^2$</td>
<td>${v_{i2}, u_{(i+1)3}}$</td>
<td>${v_{i3}}$</td>
</tr>
<tr>
<td>$v_{i3}^3$</td>
<td>${v_{i2}, u_{(i+1)3}}$</td>
<td>${v_{i3}}$</td>
</tr>
<tr>
<td>$u_{n0}$</td>
<td>$\overline{S}$</td>
<td>${u_{n0}}$</td>
</tr>
<tr>
<td>$u_{n1}$</td>
<td>${\overline{b}<em>n, x_n} \cup {u</em>{n0}, t}$</td>
<td>${u_{n1}}$</td>
</tr>
<tr>
<td>$u_{n2}$</td>
<td>${\overline{b}<em>n, x_n, u</em>{n1}} \cup \overline{S}$</td>
<td>${u_{n2}}$</td>
</tr>
<tr>
<td>$u_{n3}$</td>
<td>${u_{n2}, t}$</td>
<td>${u_{n3}}$</td>
</tr>
<tr>
<td>$v_{n1}$</td>
<td>${\overline{b}<em>n, x_n} \cup {u</em>{n0}, s_m}$</td>
<td>${v_{n1}}$</td>
</tr>
<tr>
<td>$v_{n2}$</td>
<td>${\overline{b}<em>n, x_n, v</em>{n1}} \cup \overline{S}$</td>
<td>${v_{n2}}$</td>
</tr>
<tr>
<td>$v_{n3}$</td>
<td>${u_{n2}, s_m}$</td>
<td>${v_{n3}}$</td>
</tr>
<tr>
<td>$v_{n3}^2$</td>
<td>${v_{n2}, t}$</td>
<td>${v_{n3}}$</td>
</tr>
<tr>
<td>$v_{n3}^3$</td>
<td>${v_{n2}, s_m}$</td>
<td>${v_{n3}}$</td>
</tr>
<tr>
<td>$u_{(i+1)k}^4/v_{(i+1)k}^4$</td>
<td>${b_i}$</td>
<td>${\overline{u}<em>{(i+1)k}/\overline{v}</em>{(i+1)k}}$</td>
</tr>
</tbody>
</table>

Table 4: Actions in the set $B_1 \cup \cdots \cup B_n$ for $1 \leq i < n$ and $0 \leq k \leq 3$.

Note that variables $x_1, \ldots, x_{i-1}$ are static in $P_i'(p_{ii})$ and initialized to $p_i$. Actions in $A_S$ may have preconditions on $x_1, \ldots, x_{i-1}$, which is the purpose of including these variables. We define another partial planning instance
\( P'_{i2}(p_i) \) identical to \( P'_{i1}(p_i) \) except the goal is \( v_{i3} \) instead of \( u_{i3} \).

For each \( 1 \leq i \leq n \), the partial planning instance \( P'_{i1}(p_i) \) has associated loop instance \( U_{i1} \) since \( u_{i1} \) and \( u_{i2} \) are initially false and \( u_{i2} \) is a precondition of the only action \( u_{i3} \) that adds the goal \( u_{i3} \). However, if we instead consider \( P'_{i2}(p_i) \), there are three actions that add the goal \( v_{i3} \), namely \( v_{i1} \), \( v_{i2} \), and \( v_{i3} \) with precondition \( v_{i2} \). In other words, neither \( u_{i2} \) nor \( v_{i2} \) is a landmark for \( P'_{i2}(p_i) \), but rather \( \{u_{i2}, v_{i2}\} \) is a disjunctive landmark.

For each \( 1 \leq i \leq n \) and each assignment \( p_i \), the actions are defined such that exactly one of \( P'_{i1}(p_i) \) and \( P'_{i2}(p_i) \) is solvable. Which of the two instances is solvable depends on the partial QBF \( F_i(p_i) \) and the parity of \( i \). If \( i \) is odd, \( x_i \) is universal, in which case \( P'_{i1}(p_i) \) is solvable if and only if \( F_i(p_i) \) is satisfiable and \( P'_{i2}(p_i) \) is solvable if and only if \( F_i(p_i) \) is unsatisfiable. Conversely, if \( i \) is even, \( x_i \) is existential, in which case \( P'_{i2}(p_i) \) is solvable if and only if \( F_i(p_i) \) is satisfiable and \( P'_{i1}(p_i) \) is solvable if and only if \( F_i(p_i) \) is unsatisfiable. Since \( P'_{11}(p_1) = P'_{11}(\emptyset) = P_F \) and \( i = 1 \) is odd, this implies that \( P_F \) is solvable if and only if the QBF \( F_1(p_1) = F_1(\emptyset) = F \) is satisfiable.

Figure 4 shows the causal graph of the planning instance \( P_F \). To avoid cluttering, many vertical edges have been omitted, but it is easy to verify that each edge is either left-to-right within the same row of variables, or top-to-bottom between rows of variables, implying that the causal graph is acyclic. All edges induced by the actions for \( X_i \) are already present. For \( S \), the edges not shown are those associated with the literals of each clause, i.e. each edge is from a variable among \( x_1, \ldots, x_n \) to either \( s_j \) or \( t \). The edges to \( V_i \) not shown are from \( b_{i-1} \) (for \( i > 1 \)), \( b_i \), \( x_i \), and \( V_{i+1} \) or, in the case of \( i = n \), from \( S \). Variables \( v_{11}, v_{12}, v_{13} \) do not appear since they are irrelevant.
4.3. Proof of Correctness

In this section we prove that the reduction is correct, i.e. that the planning instance $P_F$ constructed in the previous section has a solution if and only if the QBF $F$ is satisfiable. We first prove that the variables in $S$ and actions in $A_S$ correspond to the algorithm Check that tests whether the formula $\phi$ is satisfied given the current assignment $p_{n+1}$ to $x_1, \ldots, x_n$.

**Lemma 8.** Given an assignment $p_{n+1}$, starting from $S$ it is possible to set $s_m$ to true if and only if $\phi$ is satisfied, and $t$ to true if and only if $\phi$ is unsatisfied.
We show by induction on $1 \leq j \leq m$ that we can set $s_j$ to true if and only if clauses $c_1, \ldots, c_j$ are satisfied, and $t$ to true if and only if at least one of these clauses is unsatisfied. For $j = 1$, if $c_1$ is satisfied, at least one action among $s_1^1$, $s_1^2$, $s_1^3$ is applicable, but not $t^1$, making it possible to set $s_1$ to true, but not $t$. If $c_1$ is unsatisfied, action $t^1$ is applicable, but not $s_1^1$, $s_1^2$, $s_1^3$, making it possible to set $t$ to true, but not $s_1$.

For $j > 1$, if at least one clause among $c_1, \ldots, c_{j-1}$ is unsatisfied, by hypothesis of induction we can set $t$ to true but not $s_{j-1}$. Then no action among $s_j^1$, $s_j^2$, $s_j^3$ is applicable, making it impossible to set $s_j$ to true. If, on the contrary, clauses $c_1, \ldots, c_{j-1}$ are satisfied, we can set $s_{j-1}$ to true but not $t$. Then if $c_j$ is satisfied, at least one action among $s_j^1$, $s_j^2$, $s_j^3$ is applicable, but not $t^j$, making it possible to set $s_j$ to true, but not $t$. If $c_j$ is unsatisfied, action $t^j$ is applicable, but not $s_j^1$, $s_j^2$, $s_j^3$, making it possible to set $t$ to true, but not $s_j$. \qed

We next prove that, given some assignment $p_i$ to the variables $x_1, \ldots, x_{i-1}$, which of $P_i'(p_i)$ and $P_i''(p_i)$ is solvable tells us whether or not $F_i(p_i)$ is satisfiable, effectively simulating the algorithm QSat in Table 3.

**Lemma 9.** For each $1 \leq i \leq n$, let $p_i$ be an assignment to $x_1, \ldots, x_{i-1}$. The instance $P_i'(p_i)$ is solvable if and only if $i$ is odd and $F_i(p_i)$ satisfiable, or $i$ is even and $F_i(p_i)$ unsatisfiable. The instance $P_i''(p_i)$ is solvable if and only if $i$ is odd and $F_i(p_i)$ unsatisfiable, or $i$ is even and $F_i(p_i)$ satisfiable.

**Proof.** By induction on $1 \leq i \leq n$. For $i = n$, to solve $P_{n1}'(p_i)$ we have to apply the action subsequence $\langle u_{n0}^1, u_{n1}^1, u_{n2}^1, u_{n3}^1 \rangle$. Actions $u_{n0}^1$ and $u_{n2}^1$ have precondition $\overline{S}$, and actions $u_{n1}^1$ and $u_{n3}^1$ have precondition $t$. Moreover, $U_{n1}$
is a loop instance for $P'_{n_1}(p_i)$. We thus have to make $t$ true starting from $\overline{S}$ for $x_n$ false and $x_n$ true given the assignment $p_n$. Due to Lemma 8, this is possible if and only if $φ$ is unsatisfied for $p_n \cup \{\overline{x}_n\}$ and $p_n \cup \{x_n\}$. Since $n$ is even, and hence $x_n$ existential, this corresponds to $F_n(p_n)$ being unsatisfied.

On the other hand, to solve $P'_{i_2}(p_i)$ we have to apply one of the three following action subsequences, with associated precondition sequences:

$$\langle u_{n_0}^1, u_{n_1}^1, u_{n_2}^1, v_{n_3}^1 \rangle : (\overline{S}, \{t\}, \overline{S}, \{s_m\}),$$

$$\langle u_{n_0}^1, v_{n_1}^1, v_{n_2}^1, v_{n_3}^2 \rangle : (\overline{S}, \{s_m\}, \overline{S}, \{t\}),$$

$$\langle u_{n_0}^1, v_{n_1}^1, v_{n_2}^1, v_{n_3}^3 \rangle : (\overline{S}, \{s_m\}, \overline{S}, \{s_m\}).$$

The subset $U_{n_1}$ is a loop instance of the former, while $U_{n_2}$ is a loop instance of the two latter, implying that $x_n$ is false before $u_{n_1}/v_{n_1}$ and true after $u_{n_2}/v_{n_2}$.

The three sequences of preconditions are mutually exclusive since we can only make one of $s_m$ and $t$ true starting from $\overline{S}$. In all three cases, we have to make $s_m$ true starting from $\overline{S}$ for either $p_n \cup \{\overline{x}_n\}$ or $p_n \cup \{x_n\}$, which corresponds to $F_n(p_n)$ being satisfied since $x_n$ is existential.

For $1 \leq i < n$, the reasoning is similar. To solve $P'_{i_1}(p_i)$ we have to make $v_{(i+1)3}$ true starting from $\overline{V}_{i+1}$ for $p_i \cup \{\overline{x}_i\}$ and $p_i \cup \{x_i\}$, which corresponds to solving the instances $P'_{(i+1)2}(p_i \cup \{\overline{x}_i\})$ and $P'_{(i+1)2}(p_i \cup \{x_i\})$. If $i$ is odd, the induction hypothesis states that $F_{i+1}(p_i \cup \{\overline{x}_i\})$ and $F_{i+1}(p_i \cup \{x_i\})$ are satisfiable, implying that $F_i(p_i)$ is satisfiable since $x_i$ is universal. If $i$ is even, $F_{i+1}(p_i \cup \{\overline{x}_i\})$ and $F_{i+1}(p_i \cup \{x_i\})$ are unsatisfiable, implying that $F_i(p_i)$ is unsatisfiable since $x_i$ is existential.

Conversely, to solve $P'_{i_2}(p_i)$ we have to make $u_{(i+1)3}$ true starting from $\overline{V}_{i+1}$ for either $p_i \cup \{\overline{x}_i\}$ or $p_i \cup \{x_i\}$, which corresponds to solving the instance $P'_{(i+1)1}(p_i \cup \{\overline{x}_i\})$ or $P'_{(i+1)1}(p_i \cup \{x_i\})$. If $i$ is odd, the induction hypothesis
states that $F_{i+1}(p_i \cup \{x_i\})$ or $F_{i+1}(p_i \cup \{x_i\})$ is unsatisfiable, implying that $F_i(p_i)$ is unsatisfiable since $x_i$ is universal. If $i$ is even, $F_{i+1}(p_i \cup \{x_i\})$ or $F_{i+1}(p_i \cup \{x_i\})$ is satisfiable, implying that $F_i(p_i)$ is satisfiable since $x_i$ is existential.

We are now ready to prove the main result of this section.

**Theorem 1.** PE(Ayc) is PSPACE-complete.

**Proof.** Let $F$ be an arbitrary QBF on $n$ variables and $m$ clauses in prenex normal form. We can construct the planning instance $P_F$ in polynomial time given $F$. A plan solving $P_F$ simulates a nested loop on $x_1, \ldots, x_n$. Lemma 9 states that since $i = 1$ is odd, we can solve $P_{11}'(p_1) = P_{11}'(\emptyset) = P_F$ if and only if the QBF $F_1(\emptyset) = F$ is satisfiable.

We have presented a polynomial-time reduction from QBF-SAT, a known PSPACE-complete problem, to PE(Ayc). Membership in PSPACE follows from Theorem 4 of Bäckström and Nebel (1995).

As an immediate consequence of Theorem 1, bounded plan existence is also PSPACE-complete for the class Ayc.

**Corollary 10.** BPE(Ayc) is PSPACE-complete.

We remark that for STRIPS planning instances with acyclic causal graph and positive preconditions, plan existence is in P and bounded plan existence is NP-complete. These results follow from Theorems 3.7 and 4.2 of Bylander (1994), who did not mention the causal graph but nevertheless constructed planning instances whose causal graphs are acyclic.

We also prove an upper bound on the length of an optimal plan solving $P_F$, which we later need to prove PSPACE-completeness of BPE(SC-Ayc).
Lemma 11. An upper bound on the length of an optimal plan solving $P_F$ is given by $L(m, n) = (2^{n+1} - 1)m + 18 \cdot 2^n - 10n - 18$, where $m$ is the number of clauses of the QBF formula $F$, and $n$ is the number of variables of $F$.

Proof. We prove by induction on $i$ that the length of an optimal plan for $P_{i1}'(p_i)$ and $P_{i2}'(p_i)$ is upper bounded by $L(m, n + 1 - i)$, regardless of the assignment $p_i$. The base case is given by $i = n$. To solve $P_{n1}'(p_n)$ or $P_{n2}'(p_n)$ we need to apply one of four action subsequences:

$$\langle u_{n0}^1, u_{n1}^1, u_{n2}^1, u_{n3}^1 \rangle,$$
$$\langle u_{n0}^1, u_{n1}^1, u_{n2}^1, v_{n3}^1 \rangle,$$
$$\langle u_{n0}^1, v_{n1}^1, v_{n2}^1, v_{n3}^1 \rangle,$$
$$\langle u_{n0}^1, v_{n1}^1, v_{n2}^1, v_{n3}^3 \rangle.$$

The fourth sequence requires first making $s_m$ true starting from $\overline{S}$, then resetting all variables in $S$ to false, then making $s_m$ true again, for a total of $3m$ actions. Since $U_{n2} = \{a_n, b_n, x_n, v_{n1}, v_{n2}\}$ is a loop instance for this partial planning instance, we have to insert the action sequence $\langle b_n^1, a_n^1, x_n^1, b_n^2 \rangle$ between actions $v_{n1}^1$ and $v_{n2}^1$, for a total of $3m + 8 = L(m, 1)$ actions. The remaining three sequences require making $t$ true at some point instead of $s_m$, which needs a single action instead of $m$ actions, making all of them shorter.

For $i < n$, we also need to apply one of four action sequences:

$$\langle u_{i0}^1, u_{i1}^1, u_{i2}^1, u_{i3}^1 \rangle,$$
$$\langle u_{i0}^1, u_{i1}^1, u_{i2}^1, v_{i3}^1 \rangle,$$
$$\langle u_{i0}^1, v_{i1}^1, v_{i2}^1, v_{i3}^1 \rangle,$$
$$\langle u_{i0}^1, v_{i1}^1, v_{i2}^1, v_{i3}^3 \rangle.$$

Each of these sequences requires solving $P_{(i+1)1}'(p_{i+1})$ or $P_{(i+1)2}'(p_{i+1})$ twice, first for $p_{i+1} = p_i \cup \{\overline{x_i}\}$, then for $p_{i+1} = p_i \cup \{x_i\}$. Between $u_{i1}^1/v_{i1}^1$ and
we have to reset all variables in $Z'_{i+1} = \bigcup_{j=i+1}^{n}(X_j \cup V_j) \cup S$ to false and insert the action sequence $(b_1^i, a_1^i, x_1^i, b_2^i)$.

By hypothesis of induction, solving $P'_{(i+1)1}(p_{i+1})$ or $P'_{(i+1)2}(p_{i+1})$ requires at most $L(m, n - i)$ actions. Resetting all variables in $Z'_{i+1}$ to false requires at most $m + 10(n - i)$ actions: $m$ actions to reset variables in $S$ to false and, for each $i < j \leq n$, 6 actions to reset variables in $X_j$ to false and 4 actions to reset variables in $V_j$ to false. The number of actions is upper bounded by

$$2L(m, n - i) + m + 10(n - i) + 8 =$$

$$= 2 \left[(2^{n+1-i} - 1)m + 18 \cdot 2^{n-i} - 10(n - i) - 18\right] + m + 10(n - i) + 8 =$$

$$= (2^{n+2-i} - 1)m + 18 \cdot 2^{n+1-i} - 10(n - i) - 28 =$$

$$= (2^{n+2-i} - 1)m + 18 \cdot 2^{n+1-i} - 10(n + 1 - i) - 18 = L(m, n + 1 - i).$$

Since $P'_{11}(p_1) = P'_{11}(\emptyset) = P_F$, an optimal plan for $P_F$ is upper bounded by $L(m, n + 1 - 1) = L(m, n)$. 

### 4.4. PDDL Encoding

In this section we show how to encode two planning domains in PDDL: a planning domain containing instances of type $P_n$ from Section 3, simulating nested loops on $n$ variables, and a planning domain containing instances of type $P_F$, encoding instances of QBF-SAT as planning instances.

To implement a planning domain simulating nested loops, we define a single type \textit{index} as well as predicates \textit{a}, \textit{b}, \textit{x}, \textit{u}_0, \textit{u}_1, \textit{u}_2, and \textit{u}_3, each with a single parameter in the form of an index. We also need two predicates \textit{last}, on one index, and \textit{consecutive}, on two indices.

For a given $n$, the idea is to introduce objects $j_1, \ldots, j_n$ of type \textit{index}. For each $1 \leq i \leq n$, the fluent $a(j_i)$ corresponds to the variable $a_i$ of the
planning instance $P_n$, and so on. The initial state of the PDDL planning instance is given by \{last($j_n$), consecutive($j_1, j_2$), \ldots, consecutive($j_{n-1}, j_n$)\}, consisting solely of static fluents.

Each action in Table 2 has preconditions and effects on variables in $X_i \cup V_i$ for some $1 \leq i \leq n$, or on variables in consecutive sets $X_i \cup V_i$ and $X_{i+1} \cup V_{i+1}$. We parameterize actions of the first type on a single index $j_i$, and actions of the second type on two indices $j_i$ and $j_{i+1}$, using the precondition consecutive($j_i, j_{i+1}$) to ensure that the indices are consecutive. Finally, we append the precondition last($j_i$) to actions $u^1_{n0}, \ldots, u^1_{n3}$, ensuring that these actions are only applicable for index $j_n$.

In total, the resulting planning domain has 18 actions: six actions on variables in $X_i$ for $1 \leq i \leq n$, four actions making variables in $V_i$ true for $1 \leq i < n$, four actions making variables in $V_i$ true for $i = n$, and four actions resetting variables in $V_i$ to false for $1 \leq i \leq n$. A sample PDDL encoding that includes some actions of the domain appears in Table 4.4.

In experiments, the planning domain is highly challenging, which we attribute to the fact that there are a lot of deadends. LAMA 2011 (Richter et al., 2011) performs best of the planners we tested, but is only able to solve the planning instance $P_n$ for $n \leq 4$. For $n = 4$, the solution contains 200 grounded operators (the optimal plan length of $P_n$ is $16 \cdot 2^n - 10n - 16$).

The planning domain encoding QBF instances is similar to that simulating nested loops. There is a type index with associated predicates that correspond to variables in the sets $X_i$ and $V_i$, the latter containing additional variables compared to the planning instance $P_n$.

In addition to the variables in $X_i \cup V_i$, we have to represent the variables
(define (domain nested-loop)
 (:requirements :typing)
 (:types index)
 (:predicates (last ?j - index) (consecutive ?j1 ?j2 - index)
 (a ?j - index) (b ?j - index) (x ?j - index)
 (u0 ?j - index) (u1 ?j - index)
 (u2 ?j - index) (u3 ?j - index))

 (:action a1
 :parameters (?j - index)
 :effect (and (a ?j)))

 (:action a2
 :parameters (?j1 ?j2 - index)
 :precondition (and (consecutive ?j1 ?j2) (b ?j1))
 :effect (and (not (a ?j2))))

 (:action un1
 :parameters (?j - index)
 :precondition (and (last ?j) (not (b ?j))
 (not (x ?j)) (u0 ?j))
 :effect (and (u1 ?j))

 Table 5: Sample PDDL encoding simulating nested loops

34
in the set \( S \), corresponding to the clauses of the QBF formula. To do so, we introduce a second type \texttt{clause} with associated predicates \texttt{sat}, on one clause, and \texttt{unsat}, with no parameters. To represent the precondition \( \overline{S} \) of actions \( u^1_{n0}, u^1_{n2}, \) and \( v^1_{n2} \) we have to use an ADL type \texttt{forall} construct since the number of clauses may vary between QBF instances. We remark that we can get rid of the \texttt{forall} construct using our modified reduction described in the next section.

Just as for indices, we have to keep track of the last clause as well as consecutive clauses. To distinguish between indices and clauses we introduce predicates \texttt{last-index}, \texttt{consecutive-indices}, \texttt{last-clause}, and \texttt{consecutive-clauses}. For a QBF formula \( F \) on \( n \) variables and \( m \) clauses, we introduce objects \( j_1, \ldots, j_m \) of type \texttt{index} and \( c_1, \ldots, c_m \) of type \texttt{clause}, and define the initial state by indicating the last clause and index as well as consecutive clauses and indices. Once this is done, defining the actions is straightforward.

This planning domain is even more challenging than the previous one: no planner can solve the planning instance \( P_F \) associated with a satisfiable QBF \( F \) on four variables and one clause. Since our encoding requires the QBF to be in prenex normal form, the number of variables has to be a multiple of two, and with only two variables we cannot define a meaningful QBF. Consequently, it appears that our reduction from QBF-SAT to planning instances with acyclic causal graphs is impractical to implement and solve, at least using current state-of-the-art planners.

5. Bounded Number of Preconditions

Bylander (1994) showed that the problem of plan existence is \texttt{PSPACE-}
complete for STRIPS planning instances whose actions have one postcondition and unbounded number of preconditions (either positive or negative). He did not prove a result for a bounded number \( k \) of preconditions, but conjectured that plan existence falls into the polynomial hierarchy in a regular way, with the precise complexity determined by \( k \). In this section we modify our previous reduction such that actions have at most two preconditions, thus showing that plan existence is \( \text{PSPACE} \)-complete for \( k = 2 \) and proving Bylander’s conjecture to be wrong.

We first modify loop instances such that they have at most two preconditions. Given a planning instance \( P = \langle V, A, I, G \rangle \), such a modified loop instance is a subset of variables \( U = \{a, b, x, u_1, u_2, u_3, u_4\} \subseteq V \) with associated actions \( A(U) \subseteq A \) such that \( \{u_1, u_2, u_3, u_4\} \subseteq I \) and \( u_4 \) is a landmark of \( P \). In other words, variables \( u_1, \ldots, u_4 \) are initially false, and either \( u_4 \in G \)

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Action & Partial precondition & Postcondition \\
\hline
\( a^1 \) & \( \emptyset \) & \( \{a\} \) \\
\( b^1 \) & \( \{\overline{a}\} \) & \( \{b\} \) \\
\( b^2 \) & \( \{a\} \) & \( \{\overline{b}\} \) \\
\( x^1 \) & \( \{a, b\} \) & \( \{x\} \) \\
\( x^2 \) & \( \{b\} \) & \( \{\overline{x}\} \) \\
\hline
\( u_1^1 \) & \( \{\overline{b}\} \) & \( \{u_1\} \) \\
\( u_2^1 \) & \( \{\overline{x}, u_1\} \) & \( \{u_2\} \) \\
\( u_3^1 \) & \( \{x, u_2\} \) & \( \{u_3\} \) \\
\( u_4^1 \) & \( \{\overline{b}, u_3\} \) & \( \{u_4\} \) \\
\hline
\end{tabular}
\caption{The set of actions \( A(U) \) of a modified loop instance \( U \).}
\end{table}
or $u_4$ is a precondition of some action required to reach the goal state $G$. Table 6 shows the action set $A(U)$ of a modified loop instance $U$.

Unlike the original notion of loop instance, the solution to a modified loop instance is not unique: action $b_1$ may appear before or after $u_2^1$, and action $b_2$ may appear before or after action $u_3^1$. However, the key property of loop instances still holds: on any plan solving a modified loop instance, $\{b, x\}$ holds before action $u_1^1$ and $\{b, x\}$ holds after $u_4^1$. Moreover, although the initial state does not explicitly mention the variables in $\{a, b, x\}$, these have to be initially false for the modified loop instance $U$ to be solvable.

We now modify our reduction from QBF-SAT to $\text{PE}(\text{Acyc})$ by redefining the planning instance $P_F' = \langle V_F', A_F', I_F', G_F' \rangle$ such that

- $V_F' = \bigcup_{i=1}^{n} (X_i \cup V_i') \cup E' \cup S'$,
- $V_i' = \{u_{i0}, u_{i1}, u_{i2}, \ldots, u_{i9}, v_{i2}, \ldots, v_{i9}\}$ for each $1 \leq i \leq n$,
- $E' = \bigcup_{j=0}^{m} \{e_{j0}, \ldots, e_{j3}\}$,
- $S' = \{s_1, \ldots, s_m, r_2, \ldots, r_m\} \cup \bigcup_{j=1}^{m} \{t_{j1}, \ldots, t_{j3}\}$,
- $A_F' = \bigcup_{i=1}^{n} (A_i \cup B_i') \cup A_E' \cup A_S'$,
- $I_F' = \overline{V_F'}$,
- $G_F' = \{u_{19}\}$.

For each $1 \leq i \leq n$, the set $X_i = \{a_i, b_i, x_i\}$ and the action set $A_i$ on variables in $X_i$ are the same as in the original reduction.

The set $S'$ includes many more variables than the original set $S$. Variables $s_1, \ldots, s_m$ are still used to verify that each clause is satisfied for the current
assignment to \( x_1, \ldots, x_n \). For each \( 1 \leq j \leq m \), variables \( t_{j1}, \ldots, t_{j3} \) are used to verify that clause \( c_j \) is unsatisfied. Variables \( r_2, \ldots, r_m \) are used to aggregate the information regarding some clause being unsatisfied, which is necessary since the proof of unsatisfiability for each clause is a separate chain of variables. Variables in \( E' \) are used to ensure that we can only make one variable true among \( s_1, t_{11}, \ldots, t_{m1} \) while \( b_n \) is false.

Table 7 shows the actions in the sets \( A'_E \) and \( A'_S \). To help understand the mechanism behind these actions, Figure 5 shows the subgraph of the causal graph on variables in \( E' \cup S' \). Intuitively, we can no longer use a single action to check whether a given clause \( c_j \) is unsatisfied, since this would require at least three preconditions. Instead, we check each literal of \( c_j \) in turn, corresponding to the actions \( t_{j1}^1, t_{j2}^1, \) and \( t_{j3}^1 \). The intermediate states have to be different for each clause, which is why we need variables \( t_{j1}, t_{j2}, \) and \( t_{j3} \) for each clause \( c_j \).

We proceed to prove several lemmas regarding the actions in \( A'_E \) and \( A'_S \).

**Lemma 12.** While \( b_n \) is false, starting from \( E' \cup S' \) it is impossible to make a variable \( v \in E' \cup S' \) true and then reset \( v \) to false.

**Proof.** By induction on \( v \) in the topological ordering induced by the causal graph. The base case is given by \( v = e_{01} \), an antecessor of all other variables in \( E' \cup S' \). While \( b_n \) is false we can make \( e_{01} \) true using action \( e_{01}^1 \), but \( e_{01}^2 \), the only action resetting \( e_{01} \) to false, has precondition \( b_n \).

In the inductive case, by inspection of the actions we can verify that there exists a predecessor \( u \in E' \cup S' \) of \( v \) such that each action making \( v \) true has precondition \( u \), while each action making \( v \) false has precondition \( \overline{u} \). By hypothesis of induction we cannot make \( u \) true and then reset it to false,
| $e^1_{01}$ | $\{b_n\}$ | $\{e_{01}\}$ | $e^1_{01}$ | $\{\bar{b}_n\}$ | $\{e_{01}\}$ | $s^h_1$ | $\{e^h_{05}\}$ | $s_1$ |
| $e^2_{01}$ | $\{e_{01}\}$ | $\{e_{02}\}$ | $e^2_{02}$ | $\{e_{02}\}$ | $\{e_{02}\}$ | $s^4_1$ | $\{e_{05}\}$ | $\{\pi_1\}$ |
| $e^1_{(k-1)3}$ | $\{e_{(k-1)1}, \bar{e}_{(k-1)2}\}$ | $\{e_{(k-1)3}\}$ | $e^2_{(k-1)3}$ | $\{\bar{e}_{(k-1)1}, \bar{e}_{(k-1)2}\}$ | $\{\bar{e}_{(k-1)3}\}$ | $s^h_j$ | $\{e^h_{j-1}, s_{j-1}\}$ | $s_j$ |
| $e^1_{(k-1)4}$ | $\{e_{(k-1)2}, e_{(k-1)3}\}$ | $\{e_{(k-1)4}\}$ | $e^2_{(k-1)4}$ | $\{\bar{e}_{(k-1)2}, \bar{e}_{(k-1)3}\}$ | $\{\bar{e}_{(k-1)4}\}$ | $s^4_j$ | $\{\pi_{j-1}\}$ | $ar{s}_j$ |
| $e^1_{(k-1)5}$ | $\{e_{k0}, e_{(k-1)4}\}$ | $\{e_{(k-1)5}\}$ | $e^2_{(k-1)5}$ | $\{\bar{e}_{k0}, \bar{e}_{(k-1)4}\}$ | $\{\bar{e}_{(k-1)5}\}$ | $t^1_{k1}$ | $\{t^1_{k}, e_{k5}\}$ | $\{t_{k1}\}$ |
| $e^1_{k0}$ | $\{e_{(k-1)1}, \bar{e}_{(k-1)2}\}$ | $\{e_{k0}\}$ | $e^2_{k0}$ | $\{\bar{e}_{(k-1)1}, \bar{e}_{(k-1)2}\}$ | $\{\bar{e}_{k0}\}$ | $t^2_{k1}$ | $\{\bar{e}_{k5}\}$ | $\{\bar{t}_{k1}\}$ |
| $e^1_{k1}$ | $\{e_{(k-1)2}, e_{k0}\}$ | $\{e_{k1}\}$ | $e^2_{k1}$ | $\{\bar{e}_{(k-1)2}, \bar{e}_{k0}\}$ | $\{\bar{e}_{k1}\}$ | $t^1_{kl}$ | $\{\bar{t}^1_{k}, t_{k(l-1)}\}$ | $\{t_{kl}\}$ |
| $e^1_{k2}$ | $\{e_{(k-1)3}, e_{k1}\}$ | $\{e_{k2}\}$ | $e^2_{k2}$ | $\{\bar{e}_{(k-1)3}, \bar{e}_{k1}\}$ | $\{\bar{e}_{k2}\}$ | $t^2_{kl}$ | $\{\bar{t}_{k(l-1)}\}$ | $\{\bar{t}_{kl}\}$ |
| $e^1_{m3}$ | $\{e_{m1}, \bar{e}_{m2}\}$ | $\{e_{m3}\}$ | $e^2_{m3}$ | $\{\bar{e}_{m1}, \bar{e}_{m2}\}$ | $\{\bar{e}_{m3}\}$ | $r^1_p$ | $\{r_{p-1}\}$ | $\{r_p\}$ |
| $e^1_{m5}$ | $\{e_{m2}, e_{m3}\}$ | $\{e_{m5}\}$ | $e^2_{m5}$ | $\{\bar{e}_{m2}, \bar{e}_{m3}\}$ | $\{\bar{e}_{m5}\}$ | $r^2_p$ | $\{t_{p3}\}$ | $\{r_p\}$ |
| $e^3_p$ | $\{\bar{e}_{p-1}, \bar{t}_{p3}\}$ | $\{\bar{e}_p\}$ |

Table 7: Actions in a) the set $A^*_p$ for $1 \leq k \leq m$; b) the set $A^*_j$ for $2 \leq j \leq m$, $1 \leq h \leq 3$, $1 \leq k \leq m$, $2 \leq l \leq 3$, and $3 \leq p \leq m$.  

39
rendering it impossible to make $v$ true and reset it to false. 

Lemma 13. While $b_n$ is false, to make $s_m$ or $r_m$ true starting from $\overline{E} \cup \overline{S}$ we can make at most one of $e_{(j-1)3}$ and $e_j$ true for each $1 \leq j \leq m$.

Proof. To make $s_m$ or $r_m$ true starting from $\overline{E} \cup \overline{S}$ while $b_n$ is false, we have to
either make $s_1, \ldots, s_m$ true in sequence, thus verifying that the 3SAT formula $\phi$ is satisfied by the current assignment to $x_1, \ldots, x_n$, or choose a clause $c_j$ and make $t_{j1}, t_{j2}, t_{j3}$ true in sequence, thus verifying that $c_j$ is unsatisfied. In the latter case, we have to finish by making $r_j, \ldots, r_m$ true (or $r_{j+1}, \ldots, r_m$ if $j = 1$).

Making $s_1, \ldots, s_m$ true requires first making $e_{03}, e_{04}, e_{05}$ true. Due to Lemma 12 we cannot make $e_{10}$ true before applying action $e_{05}^1$ since the latter has precondition $\overline{e}_{10}$. Likewise, we cannot make $e_{10}$ true after $e_{04}^1$, since the latter has precondition $e_{02}$ and the only action $e_{10}^1$ making $e_{10}$ true has precondition $\overline{e}_{02}$. Since $e_{04}^1$ has to appear before $e_{05}^1$ to make $e_{03}, e_{04}, e_{05}$ true, this prevents us from making $e_{10}$ true at all. The same argument holds regarding $e_{(j+1)0}$ if we want to make $t_{j1}, t_{j2}, t_{j3}$ true for any $1 \leq j < m$.

Conversely, if we want to make $t_{11}, t_{12}, t_{13}$ true, we first have to make $e_{10}, e_{11}, e_{12}$ true. Due to Lemma 12 we cannot make $e_{03}$ true before applying action $e_{12}^1$ since the latter has precondition $\overline{e}_{03}$. Likewise, we cannot make $e_{03}$ true after $e_{11}^1$, since the latter has precondition $e_{02}$ and the only action $e_{03}^1$ making $e_{03}$ true has precondition $\overline{e}_{02}$. Since $e_{11}^1$ has to appear before $e_{12}^1$ to make $e_{10}, e_{11}, e_{12}$ true, this prevents us from making $e_{03}$ true at all. The same argument holds regarding $e_{(j-1)3}$ if we want to make $t_{j1}, t_{j2}, t_{j3}$ true for any $2 \leq j \leq m$.

As a consequence of Lemma 13, to make $s_m$ or $r_m$ true we can only make variables true along a single path.

**Corollary 14.** After making $s_m$ or $r_m$ true starting from $E' \cup S'$ while $b_n$ is false, there is a directed path $p$ in the causal graph from $e_{01}$ to $s_m$ or $r_m$ such that all variables on $p$ are true and all other variables in $E' \cup S'$ are false.
If the 3SAT formula $\phi$ is satisfiable, the path $p$ from Corollary 14 is $\langle e_{01}, \ldots, e_{05}, s_1, \ldots, s_m \rangle$, else there exists $1 \leq j \leq m$ such that the path is $\langle e_{01}, e_{02}, e_{10}, e_{11}, e_{12}, \ldots, e_{j0}, \ldots, e_{j5}, t_{j1}, t_{j2}, t_{j3}, \ldots, r_m \rangle$. The only other actions whose preconditions are satisfied during this process are those making a variable among $e_{03}, \ldots, e_{(j-1)3}, e_{(j+1)0}$ true. All other actions require a variable not on the path $p$ to be true. Due to Lemma 13, no variable among $e_{03}, \ldots, e_{(j-1)3}, e_{(j+1)0}$ can be made true simultaneously with the variables $e_{10}, \ldots, e_{j0}, e_{j3}$ on the path $p$.

**Lemma 15.** Assume that all variables in $E' \cup S'$ are false except those on a path $p$ from $e_{01}$ to $s_m$ or $r_m$. Starting from this situation, satisfying $\{s_m, r_m\}$ while $b_n$ is true causes all variables in $E' \cup S'$ to be false.

**Proof.** From the given situation it is easy to show that the converse of Lemma 12 holds: we cannot make a variable in $E' \cup S'$ false and then true. While $b_n$ is true, action $e_{01}^2$ making $e_{01}$ false is applicable, but not $e_{01}^1$. The inductive case is identical to that in the proof of Lemma 12. Moreover, making a variable on $p$ false requires first making its predecessor false. Since the last variable on $p$ is $s_m$ or $r_m$, making $s_m$ and $r_m$ false has the effect of making all variables on $p$ false. During this process we cannot make any variable outside $p$ true, since the precondition of actions $e_{j3}$ and $e_{(j+1)0}$ is $\{e_{j1}, e_{j2}\}$ and $e_{j1}$ has to be false before making $e_{j2}$ false. $\square$

Table 8 shows the actions in the set $B_n'$ for $1 \leq k \leq 9$. There are a number of differences with respect to the original action set $B_n$. First, the precondition $\mathcal{S}$ has been replaced with $\{\overline{s}_m, \overline{r}_m\}$, which in turn has been split across two actions (the pairs $(u_{n0}^l, u_{n1}^l)$, $(u_{n7}^l, u_{n8}^l)$, and $(u_{n7}^l, u_{n8}^l)$, respectively). Sec-
<table>
<thead>
<tr>
<th>Action</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{n0}^1$</td>
<td>${b_{n-1}, s_m}$</td>
<td>${u_{n0}}$</td>
</tr>
<tr>
<td>$u_{n1}^1$</td>
<td>${\overline{r}<em>m, u</em>{n0}}$</td>
<td>${u_{n1}}$</td>
</tr>
<tr>
<td>$u_{n2}^1$</td>
<td>${r_m, u_{n1}}$</td>
<td>${u_{n2}}$</td>
</tr>
<tr>
<td>$u_{n3}^1$</td>
<td>${\overline{b}<em>n, u</em>{n2}}$</td>
<td>${u_{n3}}$</td>
</tr>
<tr>
<td>$u_{n4}^1$</td>
<td>${x_n, u_{n3}}$</td>
<td>${u_{n4}}$</td>
</tr>
<tr>
<td>$u_{n5}^1$</td>
<td>${x_n, u_{n4}}$</td>
<td>${u_{n5}}$</td>
</tr>
<tr>
<td>$u_{n6}^1$</td>
<td>${\overline{b}<em>n, u</em>{n5}}$</td>
<td>${u_{n6}}$</td>
</tr>
<tr>
<td>$u_{n7}^1$</td>
<td>${s_m, u_{n6}}$</td>
<td>${u_{n7}}$</td>
</tr>
<tr>
<td>$u_{n8}^1$</td>
<td>${\overline{r}<em>m, u</em>{n7}}$</td>
<td>${u_{n8}}$</td>
</tr>
<tr>
<td>$u_{n9}^1$</td>
<td>${r_m, u_{n8}}$</td>
<td>${u_{n9}}$</td>
</tr>
<tr>
<td>$v_{n2}^1$</td>
<td>${s_m, u_{n9}}$</td>
<td>${v_{n2}}$</td>
</tr>
<tr>
<td>$v_{n3}^1$</td>
<td>${b_n, v_{n2}}$</td>
<td>${v_{n3}}$</td>
</tr>
<tr>
<td>$v_{n4}^1$</td>
<td>${x_n, v_{n3}}$</td>
<td>${v_{n4}}$</td>
</tr>
<tr>
<td>$v_{n5}^1$</td>
<td>${x_n, v_{n4}}$</td>
<td>${v_{n5}}$</td>
</tr>
<tr>
<td>$v_{n6}^1$</td>
<td>${\overline{b}<em>n, v</em>{n5}}$</td>
<td>${v_{n6}}$</td>
</tr>
<tr>
<td>$v_{n7}^1$</td>
<td>${s_m, v_{n6}}$</td>
<td>${v_{n7}}$</td>
</tr>
<tr>
<td>$v_{n8}^1$</td>
<td>${\overline{r}<em>m, v</em>{n7}}$</td>
<td>${v_{n8}}$</td>
</tr>
<tr>
<td>$v_{n9}^1$</td>
<td>${s_m, v_{n8}}$</td>
<td>${v_{n9}}$</td>
</tr>
<tr>
<td>$v_{n9}^2/v_{n9}^3$</td>
<td>${r_m, v_{n8}}/{s_m, v_{n8}}$</td>
<td>${v_{n9}}$</td>
</tr>
<tr>
<td>$u_{n0}^2$</td>
<td>${b_{n-1}}$</td>
<td>${u_{n0}}$</td>
</tr>
<tr>
<td>$u_{nk}^4/v_{nk}^4$</td>
<td>${\overline{u}<em>{n(k-1)}, \overline{v}</em>{n(k-1)}}$</td>
<td>${\overline{u}<em>{nk}}/{\overline{v}</em>{nk}}$</td>
</tr>
</tbody>
</table>

Table 8: Actions in the set $B'_n$ for $1 \leq k \leq 9$. See the text for explanations.
ond, action \( u_{n0}^1 \) has precondition \( \bar{b}_{n-1} \). Finally, action \( u_{n0}^2 \) has precondition \( b_{n-1} \) and, for each \( 1 \leq k \leq 9 \), the actions making \( u_{nk} \) or \( v_{nk} \) false have precondition \( \{\bar{u}_{n(k-1)}, \bar{v}_{n(k-1)}\} \) (the action \( v_{n1}^4 \) is omitted for \( k = 1 \), and the precondition \( \bar{v}_{n(k-1)} \) is omitted for \( k = 1 \) and \( k = 2 \)).

Table 9 shows the actions in the set \( B_i' \) for \( 1 \leq i < n \) and \( 1 \leq k \leq 9 \). These actions are essentially the same as those in \( B_n' \), except we have replaced the precondition \( V_i^{i+1} \) with \( \{\bar{u}_{(i+1)9}, \bar{v}_{(i+1)9}\} \) and split it across two actions. Just as before we define two partial planning instances \( P_{i1}''(p_i) \) and \( P_{i2}''(p_i) \) for each \( 1 \leq i \leq n \) and each assignment \( p_i \). The goal of \( P_{i1}''(p_i) \) is \( u_{i9} \), while the goal of \( P_{i2}''(p_i) \) is \( v_{i9} \).

In spite of the differences between \( P_F \) and \( P_F' \), the mechanism is the same: for each \( 1 \leq i \leq n \), the planning instance \( P_{i1}''(p_i) \) has associated modified loop instance \( U_{i1} = \{a_i, b_i, x_i, u_{i3}, u_{i4}, u_{i5}, u_{i6}\} \), while \( P_{i2}''(p_i) \) has alternative modified loop instances \( U_{i1} \) and \( U_{i2} = \{a_i, b_i, x_i, v_{i3}, v_{i4}, v_{i5}, v_{i6}\} \). It is easy to prove the equivalent of Lemmas 12 and 15 for the variables in \( V_i' \): while solving \( P_{i1}''(p_i) \) or \( P_{i2}''(p_i) \), we cannot make a variable in \( V_i' \) true and then false, and when subsequently making \( u_{i9} \) and \( v_{i9} \) false while \( b_{i-1} \) is true, all variables in \( V_i' \) become false.

**Theorem 2.** The planning instance \( P_F' \) has a solution if and only if the QBF formula \( F \) is satisfiable.

**Proof.** We show that the equivalent of Lemma 9 holds for partial planning instances \( P_{i1}''(p_i) \) and \( P_{i2}''(p_i) \). For \( i = n \), to solve \( P_{n1}''(p_n) \) we have to apply the action subsequence \( \langle u_{n0}^1, \ldots, u_{n9}^1 \rangle \) with associated loop instance \( U_{n1} \). This requires making \( r_m \) true starting from \( \bar{E}' \cup \bar{S}' \) while \( b_n \) is false, then satisfying \( \{\bar{s}_m, \bar{r}_m\} \) while \( b_n \) is true, then making \( r_m \) true again while \( b_n \) is
<table>
<thead>
<tr>
<th>Action</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{i0}^1$</td>
<td>${ \overline{b}<em>{i-1}, \overline{u}</em>{i+1} }$</td>
<td>${ u_{i0} }$</td>
</tr>
<tr>
<td>$u_{i1}^1$</td>
<td>${ \bar{v}<em>{i+1}, u</em>{i0} }$</td>
<td>${ u_{i1} }$</td>
</tr>
<tr>
<td>$u_{i2}^1$</td>
<td>${ v_{i+1}, u_{i1} }$</td>
<td>${ u_{i2} }$</td>
</tr>
<tr>
<td>$u_{i3}^1$</td>
<td>${ \overline{b}<em>i, u</em>{i2} }$</td>
<td>${ u_{i3} }$</td>
</tr>
<tr>
<td>$u_{i4}^1$</td>
<td>${ \overline{x}<em>i, u</em>{i3} }$</td>
<td>${ u_{i4} }$</td>
</tr>
<tr>
<td>$u_{i5}^1$</td>
<td>${ x_i, u_{i4} }$</td>
<td>${ u_{i5} }$</td>
</tr>
<tr>
<td>$u_{i6}^1$</td>
<td>${ \overline{b}<em>i, u</em>{i3} }$</td>
<td>${ u_{i6} }$</td>
</tr>
<tr>
<td>$u_{i7}^1$</td>
<td>${ \bar{u}<em>{i+1}, u</em>{i6} }$</td>
<td>${ u_{i7} }$</td>
</tr>
<tr>
<td>$u_{i8}^1$</td>
<td>${ \bar{v}<em>{i+1}, u</em>{i7} }$</td>
<td>${ u_{i8} }$</td>
</tr>
<tr>
<td>$u_{i9}^1$</td>
<td>${ v_{i+1}, u_{i8} }$</td>
<td>${ u_{i9} }$</td>
</tr>
<tr>
<td>$v_{i2}^1$</td>
<td>${ u_{i+1}, u_{i1} }$</td>
<td>${ v_{i2} }$</td>
</tr>
<tr>
<td>$v_{i3}^1$</td>
<td>${ \overline{b}<em>i, v</em>{i2} }$</td>
<td>${ v_{i3} }$</td>
</tr>
<tr>
<td>$v_{i4}^1$</td>
<td>${ \overline{x}<em>i, v</em>{i3} }$</td>
<td>${ v_{i4} }$</td>
</tr>
<tr>
<td>$v_{i5}^1$</td>
<td>${ x_i, v_{i4} }$</td>
<td>${ v_{i5} }$</td>
</tr>
<tr>
<td>$v_{i6}^1$</td>
<td>${ \overline{b}<em>i, v</em>{i5} }$</td>
<td>${ v_{i6} }$</td>
</tr>
<tr>
<td>$v_{i7}^1$</td>
<td>${ \bar{u}<em>{i+1}, v</em>{i6} }$</td>
<td>${ v_{i7} }$</td>
</tr>
<tr>
<td>$v_{i8}^1$</td>
<td>${ \bar{v}<em>{i+1}, v</em>{i7} }$</td>
<td>${ v_{i8} }$</td>
</tr>
<tr>
<td>$v_{i9}^1$</td>
<td>${ u_{i+1}, u_{i8} }$</td>
<td>${ v_{i9} }$</td>
</tr>
<tr>
<td>$v_{29}^3/v_{29}^3$</td>
<td>${ v_{i+1}, v_{i8} }/{ u_{i+1}, v_{i8} }$</td>
<td>${ v_{i9} }$</td>
</tr>
<tr>
<td>$u_{20}^2$</td>
<td>${ b_{i-1} }$</td>
<td>${ \overline{u}_{i0} }$</td>
</tr>
<tr>
<td>$u_{ik}^4/v_{ik}^4$</td>
<td>${ \overline{u}<em>{i(k-1)}, \overline{v}</em>{i(k-1)} }$</td>
<td>${ \overline{u}<em>{ik} }/{ \overline{v}</em>{ik} }$</td>
</tr>
</tbody>
</table>

Table 9: Actions in the set $B_i'$ for $1 \leq i < n$. See the text for explanations.
false. Corollary 14 implies that after making \( r \) true, there is a single path of variables in \( E' \cup S' \) that are true. Lemma 15 then implies that when satisfying \( \{ \overline{s}_m, \overline{r}_m \} \) while \( b_n \) is true, all variables in \( E' \cup S' \) become false. Making \( r \) true is possible if and only if the formula \( \phi \) is unsatisfied, implying that \( P''_{i1}(p_i) \) is solvable if and only if \( F_n(p_n) \) is unsatisfiable.

Conversely, to solve \( P''_{n2}(p_n) \) we have to apply one of three subsequences:

\[
\langle u_{n0}^1, u_{n1}^1, v_{n2}^1, \ldots, u_{n8}^1, v_{n9}^1 \rangle,
\langle u_{n0}^1, u_{n1}^1, v_{n2}^1, \ldots, v_{n8}^2, v_{n9}^2 \rangle,
\langle u_{n0}^1, u_{n1}^1, v_{n2}^1, \ldots, v_{n8}^1, v_{n9}^3 \rangle.
\]

The first of these subsequences has associated loop instance \( U_{n1} \), while the latter two have associated loop instance \( U_{n2} \). Since each subsequence requires making \( s_m \) true for \( p_n \cup \{ \overline{x}_n \} \) or \( p_n \cup \{ x_n \} \), \( P''_{n2}(p_n) \) is solvable if and only if \( F_n(p_n) \) is satisfiable.

For \( i < n \), solving \( P''_{i1}(p_i) \) or \( P''_{i2}(p_i) \) requires making \( u_{(i+1)9} \) or \( v_{(i+1)9} \) true starting from \( \overline{V}_{i+1}' \) while \( b_i \) is false, then satisfying \( \{ \overline{x}_{(i+1)9}, \overline{v}_{(i+1)9} \} \) while \( b_i \) is true, then making \( u_{(i+1)9} \) or \( v_{(i+1)9} \) true again while \( b_i \) is false. Satisfying \( \{ \overline{x}_{(i+1)9}, \overline{v}_{(i+1)9} \} \) while \( b_i \) is true causes all variables in \( V_{i+1}' \) to be false. Technically, we can apply \( u_{(i+1)0}^1 \) before the precondition \( \{ \overline{x}_{(i+1)9}, \overline{v}_{(i+1)9} \} \) is checked, but \( u_{(i+1)0}^1 \) has to appear after \( b_i^2 \), at which time the value of \( x_i \) is already fixed.

Making \( u_{(i+1)9} \) or \( v_{(i+1)9} \) true starting from \( \overline{V}_{i+1}' \) while \( b_i \) is false corresponds to solving \( P''_{(i+1)1}(p_{i+1}) \) or \( P''_{(i+1)2}(p_{i+1}) \), first for \( p_{i+1} = p_i \cup \{ \overline{x}_i \} \), then for \( p_{i+1} = p_i \cup \{ x_i \} \). We can now apply the same reasoning as in the proof of Lemma 9. Since \( P''_{11}(p_1) = P''_{11}(\varnothing) = P' \), we have shown that \( P' \) is solvable if and only if \( F_1(p_1) = F_1(\varnothing) = F \) is satisfiable. \( \square \)
6. The Complexity of Planning for ISR-Ayc

In the previous section we established that plan existence is $\textbf{PSPACE}$-complete for planning instances with acyclic causal graphs. This raises the question whether there are subclasses of Acyc for which plan existence is easier. In this section we identify one such subclass by showing that $\textbf{PE(ISR-Ayc)}$ is $\textbf{NP}$-complete. Without loss of generality we assume that the initial state of each propositional variable $v \in V$ is $\overline{v}$.

**Proposition 16.** $\textbf{PE(ISR-Ayc)}$ is $\textbf{NP}$-complete.

*Proof.* $\textbf{NP}$-hardness follows immediately from Theorem 2 in Brafman and Domshlak (2003); merely note that each variable in their construction is irreversible.

We continue by proving membership in $\textbf{NP}$. Let $P = \langle V, A, I, G \rangle$ be an arbitrary instance of ISR-Ayc, and let $V_I \subseteq V$ be the subset of irreversible variables. A plan solving $P$ cannot change the values of variables in $V_I$ more than once. We construct a non-deterministic guess as follows:

1. a subset $U \subseteq V_I$ of variables whose values change once,
2. a total order $\prec$ on $U$ (the order in which variables change),
3. for each $v \in U$, a state $s_v$ and an action $a_v$ applicable in $s_v$ that satisfies $\text{post}(a_v) = \{v\}$.

Note that the size of the guess is polynomial in the size of $P$. Given this information, we claim that we can verify whether there exists a plan solving $P$ in polynomial time. Assume for simplicity that $U = \{v_1, v_2, \ldots, v_m\}$ and
$v_1 \prec v_2 \prec \cdots \prec v_m$. A subsequence of states on a plan solving $P$ looks like

\begin{align*}
I & \rightarrow s_{v_1} \rightarrow s_{v_1} \oplus \text{post}(a_{v_1}) \rightarrow s_{v_2} \rightarrow s_{v_2} \oplus \text{post}(a_{v_2}) \rightarrow \\
& \rightarrow \ldots \rightarrow s_{v_m} \rightarrow s_{v_m} \oplus \text{post}(a_{v_m}) \rightarrow G
\end{align*}

To show that this subsequence can be extended to a valid plan, it is sufficient to show that there exists a plan from $s = s_{v_i} \oplus \text{post}(a_{v_i})$ to $s' = s_{v_{i+1}}$ for each $1 \leq i \leq m - 1$. The plan from $s$ to $s'$ is not allowed to change any irreversible variables, so we can remove all actions on $V_I$ from $A$. Let $A' = \{ a \in A : V_{\text{post}}(a) \cap V_I = \emptyset \}$ be the resulting set of actions. Then the planning instance $(V, A', s, s')$ is an instance of 3S (Jonsson and Bäckström, 1998), since the causal graph is acyclic, variables in $V_I$ are static (i.e. no action changes the variable) and all other variables are symmetrically reversible. Plan existence is in $P$ for 3S, so we can determine in polynomial time whether there exists a plan from $s$ to $s'$. Verifying that there exists a plan from $I$ to $s_{v_1}$ and from $s_{v_m} \oplus \text{post}(a_{v_m})$ to $G$ can be similarly done. \hfill \Box

7. The Complexity of Planning for SC-Acyc

In this section we study the complexity of plan existence and bounded plan existence when the causal graph is acyclic and the DTG of each variable is strongly connected. We first show that the decision problem $\text{PE}(\text{SC-Acyc})$ is in $P$ by proving that all planning instances in SC-Acyc have a solution. We then show that the decision problem $\text{BPE}(\text{SC-Acyc})$ is $\text{PSPACE}$-complete. This latter result generalizes that of Helmert (2004), who showed that bounded plan existence is $\text{NP}$-hard for the subclass of SC-Acyc with inverted fork causal graphs.
Lemma 17. For each planning instance $P$ in SC-Acyc, there exists a plan that solves $P$ (and $P_{E}(SC-Acyc)$ is in $P$).

Proof. By induction on the cardinality $|V|$. If $|V| = 1$, the resulting planning instance has a single variable, and the fact that $DTG(v)$ is strongly connected implies that we can always reach any value in $D(v)$ from any other value. Thus $P$ has a solution regardless of the values of $I$ and $G$.

If $|V| = n > 1$, choose a variable $v \in V$ without incoming edges in the causal graph $G$. Such a variable exists since $G$ is acyclic. Let $W = V \setminus \{v\}$, and let $A \mid W = \{ \langle \text{pre}(a) \mid W, \text{post}(a) \rangle : a \in A, V_{\text{post}}(a) \subseteq W \}$ be the projection of the actions in $A$ onto $W$. Compute a solution to the planning instance $\langle W, A \mid W, I \mid W, G \mid W \rangle$. Such a solution exists by induction hypothesis since $|W| < n$.

If we convert the actions in the resulting plan back to $A$, some of them might have preconditions on $v$. To compute a solution to $P$ we can now simply insert actions on $v$ that achieve these preconditions. Such actions exist since $DTG(v)$ is strongly connected and since no actions on $v$ have a precondition on other variables (else $v$ would have an incoming edge in the causal graph). If $v \in V_{G}$, also insert actions that satisfy the goal state $G(v)$ on $v$.

To prove that $BPE(SC-Acyc)$ is $PSPACE$-complete, we take advantage of our reduction from QBF-SAT to $PE(Acyc)$, described in Section 4. Given a QBF formula $F$, the planning instance $P_{F}$ that we construct has a single variable whose DTG is not strongly connected, namely $a_{1}$ (incidentally implying that a single irreversible variable is sufficient to increase the complexity of plan existence from $P$ to $PSPACE$). We modify the planning problem $P_{F}$ by
adding variables $c_1, \ldots, c_k$ where $k = \lceil \log L(m, n) \rceil$ and $L(m, n)$ is the upper bound on the length of optimal plans for $P_F$ from Lemma 11 (implying that $k$ is polynomial in the size of $F$). The variables $c_1, \ldots, c_k$ are initially false.

Table 10 shows the actions on $c_1, \ldots, c_k$, causing these variables to act as a Gray counter from 0 to $2^k - 1$ (Bäckström and Nebel, 1995). The causal graph on these variables is acyclic since there is no edge from a variable $c_i$ to a variable $c_j$ with $j < i$. We now define a single additional action $a_2$ whose precondition on $c_1, \ldots, c_k$ encodes the value $L(m, n)$, and whose effect is $\overline{a_1}$.

**Theorem 3.** The decision problem $\text{BPE}(\text{SC-Acyc})$ is $\text{PSPACE}$-complete.

**Proof.** Membership is trivial, and we prove $\text{PSPACE}$-hardness by reduction from $\text{QBF-Sat}$. Let $F$ be an arbitrary QBF formula, and let $P''_F$ be our modified planning instance from above. The causal graph of $P''_F$ is acyclic and each variable has strongly connected DTG, including $a_1$ and $c_1, \ldots, c_k$. Then Lemma 17 implies that there exists a plan solving $P''_F$.

Let $L(m, n)$ be the bound on the optimal plan length of the planning instance $P_F$ from Lemma 11. We claim that $P''_F$ has a solution of length at most $L(m, n)$ if and only if $F$ is satisfiable. If $F$ is satisfiable, the original planning instance $P_F$ has a solution plan of length at most $L(m, n)$, and this plan is also a solution to $P''_F$. If $F$ is not satisfiable, the original planning
instance $P_F$ does not have a solution. This means that we have to use action $a_1^2$ to solve $P''_F$, which requires us to first use the Gray counter $c_1, \ldots, c_k$ to count to $L(m, n)$, causing any plan solving $P''_F$ to have length greater than $L(m, n)$.

\[ \square \]

8. Related Work

The conception of the causal graph is usually credited to Knoblock (1994), who devised an algorithm that constructs abstraction hierarchies for planning instances with acyclic causal graphs. Bacchus and Yang (1994) extended this idea, improving the chance of obtaining a hierarchical solution. The causal graph heuristic (Helmert, 2004) exploits acyclic causal graphs to approximate the cost of reaching the goal. When necessary, the algorithm breaks cycles in the graph by ignoring some of the preconditions of each action.

Several authors have studied the computational complexity of planning when the causal graph is acyclic. Bäckström and Nebel (1995) showed that there are planning instances with acyclic causal graphs that have exponentially long solutions. However, this does not necessarily imply that it is hard to determine whether a solution exists (Jonsson and Bäckström, 1998). Williams and Nayak (1997) proposed a reactive planner that outputs each action in polynomial time when the causal graph is acyclic and variables are reversible. A similar algorithm was proposed by Jonsson and Bäckström (1998) for the class 3S of planning instances with acyclic causal graph and propositional variables that are either static, splitting, or symmetrically reversible. Brafman and Domshlak (2003) studied the class of planning instances with propositional variables and polytree causal graphs, and de-
signed a polynomial-time algorithm that outputs a complete solution when the causal graph has bounded indegree. Giménez and Jonsson (2008) showed that the problem of plan existence is \( \text{NP} \)-complete for this class when the indegree is unbounded. Chen and Giménez (2008) showed that when variables have domains of unbounded size, any connected causal graph containing an unbounded number of variables causes plan existence to be bounded away from \( \text{P} \).

Regarding our PDDL encoding for translating QBF formulae to planning instances, we are only aware of two previous related approaches. The first is the reduction by Bylander (1994) from deterministic Turing machine (DTM) acceptance to STRIPS planning. Although no PDDL encoding was provided, in principle we could first reduce QBF-SAT to DTM acceptance and then use Bylander’s reduction to produce a planning instance. The second approach is the work of Porco et al. (2013), who introduced a general approach to translating formulae in second order logic to planning instances in PDDL. However, this is only sufficient to translate problems in the polynomial hierarchy, not \( \text{PSPACE} \).

9. Conclusion

In this paper we have proved that the plan existence problem is \( \text{PSPACE} \)-complete when restricted to instances with acyclic causal graphs. Our proof is largely based on one conceptually simple idea: nondeterministic choices can be replaced by enumerating all possible choices. Implementing this idea in such a weak “programming language” as propositional planning is non-trivial, though, and our solution is based on making several counters to
interact in complex ways. It is not surprising that the planning instance constructed in the reduction has a causal graph that is complicated and difficult to characterise in graph-theoretical terms. Hence, it may be worthwhile to try to obtain alternative proofs that leads to instances with different (and hopefully simpler) causal graphs. An interesting question along these lines is the following: let $\text{Pe}(\mathcal{C})$ denote the plan existence problem restricted to instances such that their casual graphs are members of $\mathcal{C}$, and let $\mathcal{C}_n$ denote the directed chain on $n$ vertices. Now, is it the case that $\text{Pe}(\{\mathcal{C}_1, \mathcal{C}_2, \ldots\})$ is $\text{PSPACE}$-complete? It is known that $\text{Pe}(\{\mathcal{C}_1, \mathcal{C}_2, \ldots\})$ is NP-hard even if the variable domains are restricted to five elements (Giménez and Jonsson, 2009) but there are no results yet indicating that this problem is indeed harder.

We may take this idea one step further and try to fully characterise the sets of graphs $\mathcal{C}$ such that $\text{Pe}(\mathcal{C})$ is $\text{PSPACE}$-complete. This may appear to be an overly difficult problem but it should not be deemed completely hopeless: recall that Chen and Giménez (2008) have, under the complexity-theoretic assumption that nu-$\text{FPT} \neq \text{W}[1]$, exactly characterised the sets of graphs $\mathcal{C}$ such that $\text{Pe}(\mathcal{C})$ is in $\text{P}$. Hence, their result may be viewed as a characterisation of the problems in the “easy” end of the hardness spectrum while a characterisation of the $\text{PSPACE}$-complete problems would be a summary of the other end of the spectrum. We also note that their result leaves room for significant improvements since they only prove that sets of graphs that do not satisfy the tractability condition are not in $\text{P}$. In fact, there exists a set of graphs $\mathcal{C}$ such that $\text{Pe}(\mathcal{C})$ is NP-intermediate, i.e. $\text{Pe}(\mathcal{C})$ is not in $\text{P}$ and $\text{Pe}(\mathcal{C})$ is not NP-hard. Clearly, a characterisation of
the \textit{PSPACE}-complete graphs (and also of the $X$-complete graphs for other complexity classes $X$ within \textit{PSPACE}) would be an interesting refinement of their result.

We finally note that it may be much easier to study sets of acyclic graphs instead of general graphs. The following could be a first step: identify the sets of acyclic graphs $\mathcal{C}$ such that $\text{PE}(\mathcal{C})$ is NP-complete without imposing any other constraints on, for instance, domain sizes? Examples exist in the literature (Helmert, 2004) but they are scarce. However, recall that if we allow other side constraints (such as restricting domain sizes or otherwise put restrictions on the DTGs), then there are plenty of examples in the literature. Examples include directed-path singly connected causal graphs with domain size two (Brafman and Domshlak, 2003). Naturally, this kind of studies can be performed with other complexity classes in mind—probably, the most interesting result would be to characterise the acyclic graphs that make $\text{PE}$ tractable.

\section*{References}


55


