Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

When Acyclicity Is Not Enough: Limitations of the Causal Graph

Anders Jonsson
Dept. Information and Communication Technologies
Universitat Pompeu Fabra
Roc Boronat 138
08018 Barcelona, Spain
anders.jonsson@upf.edu

Peter Jonsson and **Tomas Lööw**
Department of Computer Science
Linköping University
SE-581 83 Linköping, Sweden
{peter.jonsson,tomas.loow}@liu.se

Abstract

Causal graphs are widely used in planning to capture the internal structure of planning instances. In the past, causal graphs have been exploited to generate hierarchical plans, to compute heuristics, and to identify classes of planning instances that are easy to solve. It is generally believed that planning is easier when the causal graph is acyclic. In this paper we show that this is not true in the worst case, proving that the problem of plan existence is **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 pre-conditions. Having established that planning is hard for acyclic causal graphs, we study a subclass of planning instances with acyclic causal graphs whose variables have strongly connected domain transition graphs. For this class, we show that plan existence is easy, but that bounded plan existence is hard, implying that optimal planning is significantly harder than satisficing planning for this class.

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.

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 derive a lower bound on the cost to reach the goal. When necessary, the algorithm breaks cycles in the graph by ignoring some of the pre-conditions 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 plan 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 designed 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 **NP**-complete for this class when the indegree is unbounded. Chen and Giménez (2008) showed that when variables have domains of unbounded size, planning is **NP**-hard for any connected causal graph, acyclic or not.

The implicit assumption common to much of the previous work has been that planning instances with acyclic causal graphs are *easier* to solve than planning instances with arbitrary causal graphs. Although plan generation may take exponential time in the worst case, the possibility has remained that plan existence is easier to determine. In this paper we prove that this assumption is false, showing that plan existence is **PSPACE**-complete for planning instances with acyclic causal graphs. This is true even if we restrict ourselves to STRIPS with negative pre-conditions. We show this result by exhibiting a reduction from QBF-SAT, i.e. the problem of deciding whether or not a quantified boolean formula is satisfiable. The reduction is based on intertwined counters that simulate a nested loop. Since plan existence is known to be **PSPACE**-complete for STRIPS planning (Bylander 1994), plan existence is no easier when the causal graph is acyclic.

We also study the problem of *optimal* planning for planning instances with acyclic causal graphs. The computational complexity of this problem has been considered in the literature but it is not as well-studied as the ordinary planning problem. Bäckström and Nebel (1995) provided complexity results for several combinations of structural restrictions, although they did not explicitly refer to the causal

graph. Helmert (2004) proved that optimal planning is **NP**-complete when the causal graph is an inverted fork, even when the domain transition graphs are strongly connected. Katz and Domshlak (2008) showed that optimal planning is tractable when the causal graph is a polytree and each action has at most one prevail-condition. Jonsson (2009) showed that optimal planning is tractable for a subclass of planning instances with tree-reducible causal graphs.

Since optimal planning is at least as hard as satisficing planning, our earlier result immediately implies that optimal planning is **PSPACE**-complete when the causal graph is acyclic. Additionally, we show that there are cases where the plan existence problem is in **P** while the optimal planning problem is computationally much harder. First, we show that if the causal graph is acyclic and the domain transition graph (DTG) of each variable is strongly connected, then the plan existence problem is trivial since *all* planning instances that belong to this class have a solution. We continue by showing that optimal planning is **#P**-hard in this case. The complexity class **#P** is a counting class and #SAT (the problem of counting the number of satisfying assignments to a SAT formula) is its canonical complete problem. One should note that the full polynomial hierarchy (and consequently **NP**) is a subset of $\mathbf{P}^{\#\mathbf{P}}$ (i.e. polynomial time with oracle access to **#P**) by Toda's theorem (Toda 1991). It is thus likely that this problem is much harder than the **NP**-complete ones. However, being **#P**-hard is not as strong as being **PSPACE**-hard; we know that **#P** is included in **PSPACE** but we do not know if this inclusion is strict. If we compare this result with the aforementioned result by Helmert (2004), then we see that lifting the restriction on the causal graph yields a considerably harder problem.

Notation

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 $v \in V_p - V_q$. Sometimes we use $(v_1 = x_1, \ldots, v_k = x_k)$ to denote a partial state p defined by $V_p = \{v_1, \ldots, v_k\}$ and $p(v_i) = x_i$ for each $v_i \in V_p$.

A planning instance is a tuple $P = \langle V, A, I, G \rangle$ where V is a set of variables, A is a set of actions, I is an initial state, and G is a (partial) goal state. Each action $a = \langle \mathrm{pre}(a), \mathrm{post}(a) \rangle \in A$ has a pre-condition $\mathrm{pre}(a)$ and a post-condition $\mathrm{post}(a)$, both partial states on V. Action a is applicable in state s if $s \mid V_{\mathrm{pre}}(a) = \mathrm{pre}(a)$, and applying a in s results in a new state $s' = s \oplus \mathrm{post}(a)$.

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 following the application of $\langle a_1, \ldots, a_{i-1} \rangle$ in I. The plan *solves* P if $s(v) = G(v)$ holds for each $v \in V_G$ in the state s that results from applying $\langle a_1, \ldots, a_k \rangle$ in I.

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 $u \in V_{\mathrm{pre}(a)} \cup V_{\mathrm{post}(a)}$ and $v \in V_{\mathrm{post}(a)}$. In this paper we focus on planning instances with acyclic causal graphs, implying that each action $a \in A$ is *unary*, i.e. satisfies $|V_{\mathrm{post}}(a)| = 1$, since two or more variables in a post-condition would induce a cycle in the causal graph.

When the variables are propositional we frequently use v and $\overline{v}$ to denote the truth value of a variable $v \in V$ instead of referring to the values in its domain $D(v)$. In this case, a partial state p can be represented as a set of *literals*, where each literal l is a positive or negative variable, i.e. $l = v$ or $l = \overline{v}$ for some $v \in V$. Given a subset of propositional variables $U \subseteq V$, we define $\overline{U} = \{\overline{v} : v \in U\}$ as the set of literals obtained by negating all variables in U.

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

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

- Acyc: planning instances with acyclic causal graphs.
- SC-Acyc: the subclass of planning instances in Acyc such that all variables have strongly connected DTGs.

Given an arbitrary planning instance P, it is easy to see that checking if P is a member of Acyc or SC-Acyc is a polynomial-time solvable task.

For each class of planning instances X, we define $\textsc{Pe}(X)$, 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 $\textsc{Bpe}(X)$, 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 $\textsc{Pe}(X)$ is polynomially reducible to $\textsc{Bpe}(X)$ since each solvable planning instance must have a solution of length at most $K = \prod_{v \in V} |D(v)|$.

PE(Acyc) is PSPACE-complete

In this section we prove that PE(Acyc) is **PSPACE**-complete by reduction from QBF-satisfiability. A quantified Boolean formula (QBF) is a conjunction of clauses such that the variables are bound by quantifiers, either existential or universal. We focus on QBFs in *prenex normal form*, i.e. the quantifiers alternate between existential and universal for each pair of variables. Let QBF-SAT be the following decision problem:

INPUT: A QBF F in prenex normal form.
QUESTION: Is F satisfiable?

```
1 function QSat(i, p_i)
2   if i = n then
3     return Check(φ, p_i ∪ {x̄_i}) | Check(φ, p_i ∪ {x_i})
4   else if i is odd then
5     return QSat(i+1, p_i∪{x̄_i}) & QSat(i+1, p_i∪{x_i})
4   else
6     return QSat(i+1, p_i∪{x̄_i}) | QSat(i+1, p_i∪{x_i})
```

Figure 1: Algorithm QSat that checks if $F_i(p_i)$ is satisfiable.

Action	Pre	Post
a^1	$\varnothing$	$\{a\}$
b^1	$\{\overline{a}\}$	$\{b\}$
b^2	$\{a\}$	$\{\overline{b}\}$
x^1	$\{b\}$	$\{x\}$
x^2	$\{b\}$	$\{\overline{x}\}$

Action	Pre	Post
u_1^1	$\{\overline{b}, \overline{x}\}$	$\{u_1\}$
u_1^2	$\varnothing$	$\{\overline{u_1}\}$
u_2^1	$\{\overline{b}, x, u_1\}$	$\{u_2\}$
u_2^2	$\varnothing$	$\{\overline{u_2}\}$

Table 1: The action sets $A(X)$ and $A(X, U)$.

The decision problem QBF-SAT is **PSPACE**-complete (Stockmeyer and Meyer 1973).

As part of reducing QBF-SAT to PE(Acyc), we first describe a general algorithm for determining whether an arbitrary QBF F is satisfiable. We then show how to construct a planning instance that simulates this algorithm. A key part of the construction is the ability to simulate nested loops using planning instances with propositional variables and acyclic causal graphs. We describe this mechanism separately since it is the most complicated part of our subsequent reduction. Moreover, simulating nested loops potentially has other uses beyond reducing QBF-SAT to PE(Acyc). We then present our reduction and prove its correctness.

QBF Satisfiability

In this section we describe an algorithm that solves the decision problem QBF-SAT in polynomial space. Let $F = \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n \cdot \phi$ be a QBF in prenex normal form, where n is an even integer, $\phi = (c_1 \wedge \cdots \wedge c_r)$, and $c_h = \ell_h^1 \vee \ell_h^2 \vee \ell_h^3$ is a 3-literal clause for each $1 \leq h \leq r$.

For each $1 \leq i \leq n$, let p_i be a partial state representing an assignment to the variables $x_1, \ldots, x_{i-1}$. Let $F_i(p_i) = Q_i x_i \cdots \forall x_{n-1} \exists x_n \cdot \phi(p_i)$ denote the QBF obtained from F by removing the quantifiers on $x_1, \ldots, x_{i-1}$ and replacing $x_1, \ldots, x_{i-1}$ in ϕ with the respective truth values in p_i. Figure 1 describes a recursive algorithm QSat that checks whether $F_i(p_i)$ is satisfiable for any arbitrary $1 \leq i \leq n$ and p_i. The algorithm Check(ϕ, p_{n+1}) returns true if and only if the 3SAT formula ϕ is satisfied by the assignment p_{n+1}, and the symbols | and & represent logical or and logical and, respectively.

Lemma 1. *The algorithm QSat runs in polynomial space and returns true if and only if $F_i(p_i)$ is satisfiable.*

Proof sketch. The recursive algorithm QSat essentially performs a nested loop on the variables $x_i, \ldots, x_n$ 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 $F_i(p_i)$ is satisfiable if and only if ϕ is satisfiable for either of the assignments $p_i \cup \{\overline{x_i}\}$ or $p_i \cup \{x_i\}$. 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. Else 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}, 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 ϕ requires $O(n + r)$ space, and the recursive calls require a total of $O(n \log n)$ space. Thus QSat runs in $O(n \log n + r)$ space, which is polynomial in F. □

Note that $F_1(p_1) = F_1(\varnothing) = F$, so Lemma 1 implies that F is satisfiable if and only if QSat$(1, \varnothing)$ returns true.

Nested Loops

Our aim is to construct a planning instance in Acyc that simulates the algorithm QSat from the previous section. To do so we first need a mechanism for simulating nested loops. There are examples in the literature of planning instances in Acyc that iterate over all assignments to n variables (Bäckström and Nebel 1995). However, none of these guarantee that assignments are not repeated, something that is crucial in our reduction. For this reason we have to devise a novel mechanism for simulating nested loops.

Let $X = \{a, b, x\}$ and $U = \{u_1, u_2\}$ be two sets of propositional variables. Given X and U, let $A(X)$ and $A(X, U)$ be the two action sets defined in Table 1. Variable x is the one whose value we wish to iterate over, and the actions affecting a variable $v \in V$ are denoted v^1, v^2, etc. Consider the planning instance $P = \langle X \cup U, A(X) \cup A(X, U), \overline{X} \cup \overline{U}, \{u_2\} \rangle$.

To achieve u_2 starting from $\overline{X} \cup \overline{U}$ we have to apply the partial action sequence $\langle u_1^1, u_2^1 \rangle$: u_2^1 to achieve u_2, and u_1^1 to achieve the pre-condition u_1 of u_2^1. The pre-condition $\{\overline{b}, \overline{x}\}$ of u_1^1 is satisfied in $\overline{X}$. To achieve the pre-condition $\{\overline{b}, x\}$ of u_2^1 we have to apply the action sequence $\langle b^1, x^1, a^1, b^2 \rangle$: x^1 to make x true, b^1 to achieve the pre-condition b of x^1, b^2 to reset b to false, and a^1 to achieve the pre-condition a of b^2. Summarizing, a plan solving P is $\langle u_1^1, b^1, x^1, a^1, b^2, u_2^1 \rangle$, resulting in the partial state $\{a, \overline{b}, x\}$ on X.

A plan solving P is highly constrained: only actions x^1 and a^1 could be swapped around, while the order of the remaining actions is strict. In the following lemma we prove a key property of plans solving P, parametrized on X and U so that we can later apply it to other sets of variables:

Lemma 2. *Given X, U, $A(X)$, and $A(X, U)$, no action sequence achieving u_2 starting from $\overline{X} \cup \overline{U}$ can change the value of a variable in X before u_1^1 or after u_2^1.*

Proof. In the partial state $\{a, \overline{b}, x\}$, no action changing the value of a variable in X is applicable. The only two such actions are b^1, with pre-condition $\overline{a}$, and x^2, with pre-condition

Action	Pre	Post
a_2^2	$\{b_1\}$	$\{\overline{a_2}\}$
u_{10}^1	$\varnothing \cup \overline{X_2} \cup \overline{U_2}$	$\{u_{10}\}$
u_{11}^1	$\{\overline{b_1}, \overline{x_1}, u_{10}\} \cup \{u_{22}\}$	$\{u_{11}\}$
u_{12}^1	$\{\overline{b_1}, x_1, u_{11}\} \cup \overline{X_2} \cup \overline{U_2}$	$\{u_{12}\}$
u_{13}^1	$\{u_{12}\} \cup \{u_{22}\}$	$\{u_{13}\}$

Table 2: The set of actions B of the planning instance P_2.

b. Since any action sequence starting in $\overline{X} \cup \overline{U}$ has to achieve the partial state $\{a, \overline{b}, x\}$ before applying u_2^1, it cannot change the value of a variable in X after u_2^1.

If a^1 appears before u_1^1, b^1 is no longer applicable, making it impossible to achieve the pre-condition $\{\overline{b}, x\}$ starting from $\{\overline{b}, \overline{x}\}$ as required by u_1^1 and u_2^1. Applying b^1 before u_1^1 requires b^2 to satisfy the pre-condition $\overline{b}$ of u_1^1, which in turn requires a^1. Finally, action x^1 has pre-condition b, which can only be achieved by b^1. Consequently, no action changing the value of a variable in X can appear before u_1^1. □

Due to Lemma 2, a plan solving P simulates a loop on x such that x is false before u_1^1 and true after u_2^1. As we will show, we can add additional actions whose pre-conditions control the inner structure of the loop. Actions x^2, u_1^2, and u_2^2 are not strictly needed to solve P; their purpose is to show that Lemma 2 still holds when they are present.

We next show how to simulate a nested loop on two variables x_1 and x_2. Let $V = X_1 \cup U_1 \cup X_2 \cup U_2$, where $X_i = \{a_i, b_i, x_i\}$ for each $1 \leq i \leq 2$, $U_1 = \{u_{10}, u_{11}, u_{12}, u_{13}\}$, and $U_2 = \{u_{21}, u_{22}\}$. Let $A = A(X_1) \cup A(X_2) \cup A(X_2, U_2) \cup B$ be a set of actions on V, where $A(X_1)$, $A(X_2)$, and $A(X_2, U_2)$ are defined as in Table 1 and B is defined in Table 2. For clarity, the pre-conditions of $u_{10}^1, \ldots, u_{10}^3$ are divided into two parts, one that controls the loop over x_1, and one that controls the body of the loop. Consider the planning instance $P_2 = \langle V, A, \overline{V}, \{u_{13}\}\rangle$.

Although U_1 contains two variables u_{10} and u_{13} not present in $U = \{u_1, u_2\}$, the structure of the actions on U_1 in B is similar to $A(X, U)$: to achieve the goal u_{13} starting from $\overline{X_1} \cup \overline{U_1}$ we have to apply the partial action sequence $\langle u_{10}^1, u_{11}^1, u_{12}^1, u_{13}^1\rangle$, and the pre-conditions $\{\overline{b_1}, \overline{x_1}\}$ of u_{11}^1 and $\{\overline{b_1}, x_1\}$ of u_{12}^1 are the same as in $A(X, U)$. We can thus apply Lemma 2 to the problem of achieving u_{13} starting from $\overline{X_1} \cup \overline{U_1}$, implying that no action that changes the value of a variable in X_1 can appear before u_{11}^1 or after u_{12}^1.

The pre-conditions of u_{10}^1 and u_{11}^1 require us to achieve u_{22} starting from $\overline{X_2} \cup \overline{U_2}$. Assuming that the actions for X_2 and U_2 are given by $A(X_2)$ and $A(X_2, U_2)$, Lemma 2 applies to this problem, implying that no action that changes the value of a variable in X_2 can appear before u_{21}^1 or after u_{22}^1. However, the set B contains an additional action a_2^2 on X_2, threatening the validity of Lemma 2. Fortunately, the action sequence achieving u_{22} has to appear before action u_{11}^1. Since $\overline{b_1}$ holds in the initial state and since no action changing the value of b_1 can appear before u_{11}^1, b_1 has to be false when achieving u_{22}, violating the pre-condition of a_2^2.

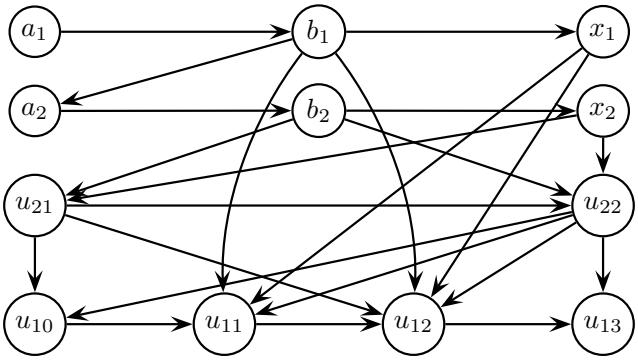

Figure 2: The causal graph of the planning instance P_2.

Action u_{12}^1 requires us to reset the variables in $X_2 \cup V_2$ to false, while u_{13}^1 again requires us to achieve u_{22}. The reset actions u_{21}^2 and u_{22}^2 in $A(X_2, U_2)$ are applicable in any state. When b_1 is true, we can apply the action sequence $\langle a_2^2, b_2^1, x_2^2, a_2^1, b_2^2, a_2^2\rangle$ to reset the variables in X_2 to false. The partial action sequence achieving u_{22} the second time has to appear after action u_{12}^1, so a_2^2 is again inapplicable since $\overline{b_1}$ holds when we apply u_{12}^1 and no action changing the value of b_1 can appear after u_{12}^1.

Summarizing, a plan solving P_2 is of the form $\langle u_{10}^1, \pi_2, u_{11}^1, b_1^1, \rho_2, x_1^1, a_1^1, b_1^2, u_{12}^1, \pi_2, u_{13}^1\rangle$, where π_2 is an action sequence achieving u_{22} starting from $\overline{X_2} \cup \overline{U_2}$, and ρ_2 is an action sequence resetting the variables in X_2 and V_2 to false. Some of the actions can be moved around, but the two instances of π_2 have to remain before u_{11}^1 and after u_{12}^1.

The action sequence π_2 simulates a loop on x_2 that first sets x_2 to false and then to true. On the other hand, a plan solving P_2 simulates a loop on x_1 that first sets x_1 to false and then to true. Since the plan contains one instance of π_2 for x_1 false and one instance of π_2 for x_1 true, the plan effectively simulates a nested loop on x_1 and x_2.

Figure 2 shows the causal graph of the planning instance P_2. For clarity, the edges from variables in X_2 to u_{10} and u_{12} have been omitted. All edges are left-to-right within the same row of variables, or top-to-bottom between different rows of variables, implying that the causal graph is acyclic.

It is straightforward to extend the idea to simulate a nested loop on n variables. The idea is to introduce action pairs such as (u_{10}^1, u_{11}^1) and (u_{12}^1, u_{13}^1) in P_2 whose pre-conditions control the inner structure of each loop. By carefully choosing these pre-conditions we ensure that Lemma 2 holds for each loop, forcing a plan to iterate over all combinations of values to solve the planning instance.

Construction

In this section we describe how to construct the planning instance in Acyc that corresponds to an arbitrary QBF formula. Let $F = \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n \cdot \phi$ be a QBF in prenex normal form with $\phi = (c_1 \wedge \cdots \wedge c_r)$ and $c_h = \ell_h^1 \vee \ell_h^2 \vee \ell_h^3$ for each $1 \leq h \leq r$. Given F, we construct a planning instance $P = \langle V, A, I, G\rangle$ where

Action	Pre	Post
s_h^1	$\{\ell_h^1, s_{h-1}\}$	$\{s_h\}$
s_h^2	$\{\ell_h^2, s_{h-1}\}$	$\{s_h\}$
s_h^3	$\{\ell_h^3, s_{h-1}\}$	$\{s_h\}$
s_h^4	$\{\}$	$\{\overline{s_h}\}$
t^h	$\{\overline{\ell_h^1}, \overline{\ell_h^2}, \overline{\ell_h^3}\}$	$\{t\}$
t^{r+1}	$\{\}$	$\{\overline{t}\}$

Table 3: The actions in the set A_S for $1 \leq h \leq r$. The pre-condition s_{h-1} of s_h^1, s_h^2, and s_h^3 is omitted for $h = 1$.

- $V = \bigcup_{i=1}^{n} (X_i \cup U_i) \cup S$,
- $X_i = \{a_i, b_i, x_i\}$ for each $1 \leq i \leq n$,
- $U_i = \{u_{i0}, u_{i1}, v_{i1}, u_{i2}, v_{i2}, u_{i3}, v_{i3}\}$ for each $1 \leq i \leq n$,
- $S = \{s_1, \ldots, s_r, t\}$,
- $A = \bigcup_{i=1}^{n} A(X_i) \cup B \cup A_S$,
- $I = \overline{V}$ and $G = \{u_{13}\}$.

The purpose of the variable set $\bigcup_{i=1}^{n} (X_i \cup U_i)$ and action set $\bigcup_{i=1}^{n} A(X_i) \cup B$ is to simulate a nested loop on the variables $x_1, \ldots, x_n$ of the QBF F. Simultaneously, the variables in U_i keep track of whether or not each $F_i(p_i)$ is satisfiable. For each assignment p_{n+1} to $x_1, \ldots, x_n$, the variable set S and action set A_S test whether ϕ is satisfied.

Table 3 shows the actions in the set A_S. The pre-condition s_{h-1} of actions s_h^1, s_h^2, and s_h^3 is omitted for $h = 1$. For each $1 \leq h \leq r$, literals ℓ_h^1, ℓ_h^2, and ℓ_h^3 should be replaced with the corresponding variable among $x_1, \ldots, x_n$, appropriately negated. To make s_h true we first have to make s_{h-1} true. The actions are defined such that we can achieve s_r starting from $\overline{S}$ if and only if ϕ is satisfied by the current assignment to $x_1, \ldots, x_n$, and we can achieve t if and only if ϕ is unsatisfied. Each variable can be reset to false in any state.

Table 4 shows the actions in the set B. For each $1 \leq i < n$, action a_{i+1}^2 allows resetting a_{i+1} to false whenever b_i is true. For each $1 \leq i \leq n$, B also contains actions with empty pre-conditions resetting each variable in U_i to false; we omit these action definitions to save space. Just as before, we divide pre-conditions into two parts: one that controls the loop and one that controls the body of the loop. For each $1 \leq i \leq n$, the actions on U_i are similar to those in $A(X, U)$, but four distinct action sequences are possible:

$\langle u_{i0}^1, u_{i1}^1, u_{i2}^1, u_{i3}^1 \rangle$, making u_{13} true,
$\langle u_{i0}^1, u_{i1}^1, u_{i2}^1, v_{i3}^1 \rangle$, making v_{13} true,
$\langle u_{i0}^1, v_{i1}^1, v_{i2}^1, v_{i3}^2 \rangle$, making v_{13} true,
$\langle u_{i0}^1, v_{i1}^1, v_{i2}^1, v_{i3}^3 \rangle$, making v_{13} true.

All four sequences have the same pre-conditions on X_i, causing Lemma 2 to apply to each of them. Just like in the planning instance P_2, the pre-conditions also require achieving $u_{(i+1)3}$ or $v_{(i+1)3}$ twice starting from $\overline{X_{i+1} \cup U_{i+1}}$, causing a plan solving P to simulate a nested loop on $x_1, \ldots, x_n$. The inner loop is defined by the actions on U_n, which are identical to those for U_i, $i < n$, except the pre-conditions require achieving s_r or t twice starting from $\overline{S}$.

Action	Pre	Post
a_{i+1}^2	$\{b_i\}$	$\{\overline{a_{i+1}}\}$
u_{i0}^1	$\varnothing \cup \overline{X_{i+1}} \cup \overline{U_{i+1}}$	$\{u_{i0}\}$
u_{i1}^1	$\{\overline{b_i}, \overline{x_i}, u_{i0}\} \cup \{v_{(i+1)3}\}$	$\{u_{i1}\}$
v_{i1}^1	$\{\overline{b_i}, \overline{x_i}, u_{i0}\} \cup \{u_{(i+1)3}\}$	$\{v_{i1}\}$
u_{i2}^1	$\{\overline{b_i}, x_i, u_{i1}\} \cup \overline{X_{i+1}} \cup \overline{U_{i+1}}$	$\{u_{i2}\}$
v_{i2}^1	$\{\overline{b_i}, x_i, v_{i1}\} \cup \overline{X_{i+1}} \cup \overline{U_{i+1}}$	$\{v_{i2}\}$
u_{i3}^1	$\{u_{i2}\} \cup \{v_{(i+1)3}\}$	$\{u_{i3}\}$
v_{i3}^1	$\{u_{i2}\} \cup \{u_{(i+1)3}\}$	$\{v_{i3}\}$
v_{i3}^2	$\{v_{i2}\} \cup \{v_{(i+1)3}\}$	$\{v_{i3}\}$
v_{i3}^3	$\{v_{i2}\} \cup \{u_{(i+1)3}\}$	$\{v_{i3}\}$
u_{n0}^1	$\varnothing \cup \overline{S}$	$\{u_{n0}\}$
u_{n1}^1	$\{\overline{b_n}, \overline{x_n}, u_{n0}\} \cup \{t\}$	$\{u_{n1}\}$
v_{n1}^1	$\{\overline{b_n}, \overline{x_n}, u_{n0}\} \cup \{s_r\}$	$\{v_{n1}\}$
u_{n2}^1	$\{\overline{b_n}, x_n, u_{n1}\} \cup \overline{S}$	$\{u_{n2}\}$
v_{n2}^1	$\{\overline{b_n}, x_n, v_{n1}\} \cup \overline{S}$	$\{v_{n2}\}$
u_{n3}^1	$\{u_{n2}\} \cup \{t\}$	$\{u_{n3}\}$
v_{n3}^1	$\{u_{n2}\} \cup \{s_r\}$	$\{v_{n3}\}$
v_{n3}^2	$\{v_{n2}\} \cup \{t\}$	$\{v_{n3}\}$
v_{n3}^3	$\{v_{n2}\} \cup \{s_r\}$	$\{v_{n3}\}$

Table 4: The actions in the set B for $1 \leq i < n$.

The four action sequences for U_i are mutually exclusive. At each iteration, the applicable sequence is determined by whether we can achieve $u_{(i+1)3}$ (s_r) or $v_{(i+1)3}$ (t) starting from $\overline{X_{i+1}} \cup \overline{U_{i+1}}$ ($\overline{S}$) for $\overline{x_i}$ and x_i. Although the action definitions for U_i are symmetric, the *meaning* of the variables is different for universal and existential variables. For odd i, x_i is universal, and if the QBF $F_i(p_i)$ is satisfiable we can make u_{i3} true starting from $\overline{X_i} \cup \overline{U_i}$, else v_{i3}. For even i, x_i is existential, and if $F_i(p_i)$ is satisfiable we can make v_{i3} true, else u_{i3}. For $i = 1$, variables v_{11}, v_{12}, v_{13} and their actions can be omitted since making v_{13} true is possible if and only if we cannot reach the goal state u_{13}.

Figure 3 shows the causal graph of the planning problem P. We have omitted many vertical edges, but it is easy to verify that all edges are left-to-right within the same row of variables, or top-to-bottom between different rows of variables, implying that the causal graph is acyclic. All edges induced by the actions for X_i are already present in the graph. 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_h or t. The edges to U_i not shown are from b_i, x_i, and U_{i+1} or, in the case of $i = n$, from S.

Proof

We proceed to prove that the planning problem P has a solution if and only if the formula F is satisfiable. We first show that the variables in S and actions in A_S correspond to the algorithm Check that tests whether the formula ϕ is satisfied

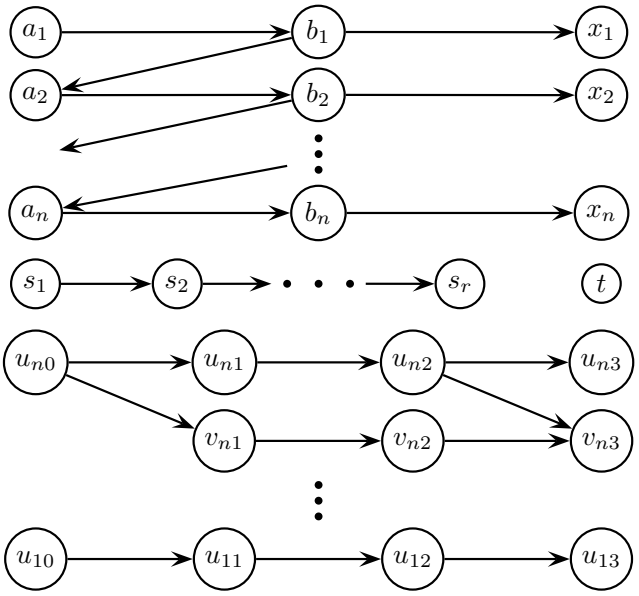

Figure 3: The causal graph of the planning problem P. For clarity, many vertical edges have been omitted.

given the current assignment p_{n+1} to $x_1, \ldots, x_n$.

Lemma 3. *Given an assignment p_{n+1} to $x_1, \ldots, x_n$, starting in $\overline{S}$ it is possible to set s_r to true iff ϕ is satisfied, and t to true iff ϕ is unsatisfied.*

Proof. By induction on $1 \leq h \leq r$. For $h = 1$, actions s_1^1, s_1^2, s_1^3, t^1 are such that it is possible to set s_1 to true iff clause c_1 is satisfied, and t to true iff c_1 is unsatisfied. For $h > 1$, the induction hypothesis states that we can set s_{h-1} to true iff clauses $c_1, \ldots, c_{h-1}$ are satisfied, and t to true iff at least one clause is unsatisfied. If s_{h-1} is true, actions s_h^1, s_h^2, s_h^3, t^h are such that we can set s_h to true iff c_h is satisfied, and t to true iff c_h is unsatisfied. □

We next prove that, given some assignment p_i to the variables $x_1, \ldots, x_{i-1}$, making u_{i3} or v_{i3} true starting from $\overline{X_i} \cup \overline{U_i}$ tells us whether or not $F_i(p_i)$ is satisfiable, effectively simulating the algorithm QSat in Table 1.

Lemma 4. *For each $1 \leq i \leq n$, given an assignment p_i to $x_1, \ldots, x_{i-1}$, starting from $\overline{X_i} \cup \overline{U_i}$ we can make u_{i3} true iff i is odd and $F_i(p_i)$ is satisfiable or i is even and $F_i(p_i)$ is unsatisfiable, else we can make v_{i3} true.*

Proof. By induction on $1 \leq i \leq n$. For $i = n$, we can make u_{n3} true iff we can make t true starting from $\overline{S}$ for $\overline{x_n}$ and x_n. Due to Lemma 3, this corresponds to ϕ being unsatisfied, which causes $F_n(p_n)$ to be unsatisfiable. We can make v_{n3} true iff we can make s_r true starting from $\overline{S}$ for either $\overline{x_n}$ or x_n, which corresponds to ϕ being satisfied, implying that $F_n(p_n)$ is satisfiable since x_n is existential.

For $1 \leq i < n$, we can make u_{i3} true iff we can make $v_{(i+1)3}$ true starting from $\overline{X_{i+1}} \cup \overline{U_{i+1}}$ for $\overline{x_i}$ and x_i. If i is odd, the induction hypothesis states that $F_{i+1}(p_i \cup \{x_i\})$ and $F_{i+1}(p_i \cup \{\overline{x_i}\})$ are satisfiable, implying that $F_i(p_i)$ is satisfiable since x_i is universal. If i is even, $F_{i+1}(p_i \cup \{x_i\})$ and $F_{i+1}(p_i \cup \{\overline{x_i}\})$ are unsatisfiable, implying that $F_i(p_i)$ is unsatisfiable since x_i is existential.

Conversely, we can make v_{i3} true iff we can make $u_{(i+1)3}$ true starting from $\overline{X_{i+1}} \cup \overline{U_{i+1}}$ for either $\overline{x_i}$ or x_i. If i is odd, the induction hypothesis states that $F_{i+1}(p_i \cup \{x_i\})$ or $F_{i+1}(p_i \cup \{\overline{x_i}\})$ is unsatisfiable, implying that $F_i(p_i)$ is unsatisfiable since x_i is universal. If i is even, by induction $F_{i+1}(p_i \cup \{x_i\})$ or $F_{i+1}(p_i \cup \{\overline{x_i}\})$ 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*(Acyc) is* ***PSPACE****-complete.*

Proof. Let F be an arbitrary QBF on n variables and r clauses in prenex normal form. We can construct the planning instance P in polynomial time given F. A plan solving P simulates a nested loop on $x_1, \ldots, x_n$. Lemma 4 states that since $i = 1$ is odd, we can make u_{13} true starting from $\overline{X_1} \cup \overline{U_1}$ if and only if the QBF $F_1(\varnothing) = F$ is satisfiable, implying that P has a solution if and only if F is satisfiable.

We have given a polynomial-time reduction from QBF-SAT, a **PSPACE**-complete problem, to PE(Acyc). Membership in **PSPACE** follows from the fact that each planning instance has a solution of length at most $\prod_{v \in V} |D(v)|$. □

Corollary 5. BPE*(Acyc) is* ***PSPACE****-complete.*

Proof. Hardness follows from Theorem 1 and the fact that PE(Acyc) is polynomially reducible to BPE(Acyc). Membership follows from the same argument as before. □

BPE(SC-Acyc) is #P-hard

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 PE(SC-Acyc) is in **P** by proving that *all* planning instances in SC-Acyc have a solution. We then show that the decision problem BPE(SC-Acyc) is **#P**-hard, which implies that BPE(SC-Acyc) is hard for the polynomial hierarchy **PH**. Note that we, as is customary, define **#P**-hardness with respect to polynomial-time Turing reductions: we say that a problem X is **#P**-hard if and only if $\#\mathbf{P} \subseteq \mathbf{P}^X$. Our result generalizes that of Helmert (2004), who showed that bounded plan existence is **NP**-hard for the subclass of SC-Acyc with inverted fork causal graphs.

Lemma 6. *For each planning instance P in SC-Acyc, there exists a plan that solves P (and* PE*(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 $W = V - \{v\}$, and let $A \mid W =$

Action	Pre	Post
s^v_{ij}	$(s = s^c_j)$	$(s = s^v_i)$
s^c_{ij}	$(s = s^v_i)$	$(s = s^c_j)$
v^1_i	$V_{i-1} \oplus (s = s^v_i)$	$(v_i = 0)$
v^2_i	$V_{i-1} \oplus (s = s^v_i)$	$(v_i = 1)$
c^1_{ik}	$C_{i-1} \oplus (s = s^c_i, \overline{\ell^1_k}, \overline{\ell^2_k}, \overline{\ell^3_k})$	$(c_i = 0)$
c^2_{ik}	$C_{i-1} \oplus (s = s^c_i, \ell^1_k, \overline{\ell^2_k}, \overline{\ell^3_k})$	$(c_i = 1)$

Table 5: The actions of the planning instance P.

$\{\langle \mathrm{pre}(a) \mid W, \mathrm{post}(a)\rangle : a \in A, V_{\mathrm{post}}(a) \subseteq W\}$ 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 pre-conditions on v. To compute a solution to P we can now simply insert actions on v that achieve these pre-conditions. Such actions exist since $DTG(v)$ is strongly connected and since no actions on v have a pre-condition 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. □

We now turn to the problem of *optimal* planning for the class SC-Acyc. We first define the decision problem ATLEAST-UNSAT as follows:

INPUT: A 3SAT formula F and an integer K.
QUESTION: Is the formula F unsatisfied by at least K distinct variable assignments?

We show that ATLEAST-UNSAT is polynomially reducible to BPE(SC-Acyc). Given an arbitrary 3SAT formula F with r clauses and n variables and an integer K, construct a planning instance $P = \langle V, A, I, G\rangle$ with

- $V = \{s, v_1, \ldots, v_n, c_1, \ldots, c_n\}$,
- $D(s) = \{s^v_1, \ldots, s^v_n, s^c_1, \ldots, s^c_n\}$,
- $D(v_i) = D(c_i) = \{0, 1\}$ for each $1 \leq i \leq n$,
- $I = (s = s^c_1, v_1 = 0, \ldots, v_n = 0, c_1 = 0, \ldots, c_n = 0)$,
- $G = (v_1 = 0, \ldots, v_{n-1} = 0, v_n = 1) \oplus$
  $\oplus\, (c_1 = x_1, \ldots, c_n = x_n)$.

Variables $v_1, \ldots, v_n$ and $c_1, \ldots, c_n$ act as two Gray counters. For each $1 \leq i \leq n$, we define the partial state $V_i = (v_1 = 0, \ldots, v_{i-1} = 0, v_i = 1)$ and the partial state $C_i = (c_1 = 0, \ldots, c_{i-1} = 0, c_i = 1)$ that are part of the pre-condition of actions that change the values of these variables. Variables $v_1, \ldots, v_n$ represent the variables of the formula F, and the goal state V_n implies that the counter has to iterate from 0 to $2^n - 1$, thus enumerating all possible assignments to $v_1, \ldots, v_n$. The values $x_1, \ldots, x_n$ in the goal state are such that the partial state $(c_1 = x_1, \ldots, c_n = x_n)$ encodes the integer K for the counter $c_1, \ldots, c_n$.

Table 5 shows the actions of the planning instance P, where the indices are in the ranges $1 \leq i \leq n$, $1 \leq j \leq n$, and $1 \leq k \leq r$. The actions on variable s are such that

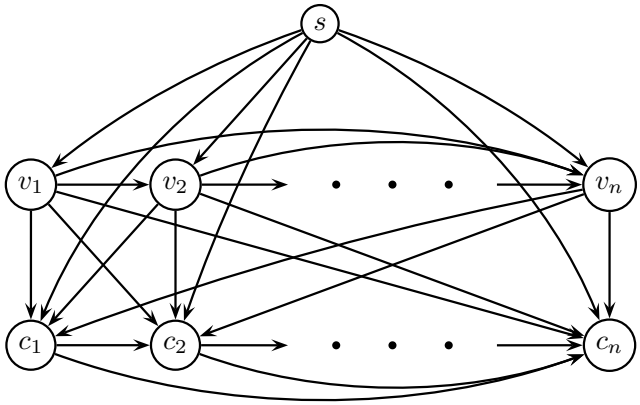

Figure 4: The causal graph of the planning instance P.

it always alternates between "v-values" (s^v_i) and "c-values" (s^c_j). The actions on $v_1, \ldots, v_n$ are those associated with a Gray counter (Bäckström and Nebel 1995), but with an additional pre-condition on s, whose "v-value" must be appropriately set. The actions on $c_1, \ldots, c_n$ are also those associated with a Gray counter, and require that the corresponding "c-value" of s is set. However, these actions also require that at least one clause is unsatisfied by the current assignment to $v_1, \ldots, v_n$. For each $1 \leq k \leq r$, the literal ℓ^1_k should be replaced by a pre-condition $v_i = 0$ or $v_i = 1$, and so on.

Lemma 7. *As long as F is not a tautology, P is in SC-Acyc.*

Proof. Figure 4 shows the causal graph of the planning instance P. All edges are either left-to-right within the same row of variables, or top-to-bottom between different rows of variables, implying that the causal graph is acyclic.

For variable s, $DTG(s)$ is strongly connected since the actions allow us to move from any "v-value" to any "c-value" and vice versa. For each $1 \leq i \leq n$, $DTG(v_i)$ is also strongly connected since actions v^1_i and v^2_i are from 0 to 1 and from 1 to 0, respectively. Intuitively, this means that the Gray counter represented by $v_1, \ldots, v_n$ can count both up and down, meaning that we can move freely between any two assignments to $v_1, \ldots, v_n$.

The Gray counter represented by $c_1, \ldots, c_n$ can also count both up and down. However, the pre-condition $(\overline{\ell^1_k}, \overline{\ell^2_k}, \overline{\ell^3_k})$ is unsatisfiable if clause k is a tautology, which in effect means that the corresponding action can never be applied. As long as there exists at least one clause that is not a tautology, there is at least one pair of actions c^1_{ik} and c^2_{ik} such that $DTG(c_i)$ is strongly connected for each $1 \leq i \leq n$. □

Lemma 8. ATLEAST-UNSAT *is polynomially reducible to* BPE*(SC-Acyc).*

Proof. Let (F, K) be an arbitrary instance of ATLEAST-UNSAT. We can construct the planning instance P in polynomial time given F and K. The corresponding instance of BPE(SC-Acyc) is given by (P, N_A), where $N_A = 3 \cdot 2^n - 3 + K$ and n is the number of variables of the formula F.

We first note that it follows by the properties of the Gray counter that any plan needs at least $3 \cdot 2^n - 4$ actions for $v_1, \ldots, v_n$ to reach the goal state V_n: $2^n - 1$ actions for incrementing the counter and $2(2^n - 1) - 1 = 2 \cdot 2^n - 3$ actions for setting the appropriate "v-value" of s. Since the second Gray counter then has at most $K+1$ steps to count K times in order to get its variables to the correct state we can conclude that that neither counter can ever count backwards since we could not have enough steps left to get to the goal state. The extra step might be needed to set s to s^c_j in case the formula is unsatisfied by the last assignment to $v_1, \ldots, v_n$.

Note also that the second counter cannot count the same unsatisfying assignment twice since it is limited to changing the value of one variable at a time, so to count an assignment twice we would have to change s to some s^v_i and then to some s^c_j before we count up again. Doing this takes 2 extra steps, but this is impossible since the plan length guarantees that we have at most $K + 1$ steps to count K times. □

We remark that due to Lemma 7, the reduction from ATLEAST-UNSAT to BPE(SC-Acyc) is only valid when F is not a tautology. However, given an arbitrary 3SAT formula F, we can check (in polynomial time) whether or not it is a tautology: simply verify that each clause in F contains some variable together with its negation. If F is indeed a tautology then F has at least K unsatisfying assignments for each $0 \leq K \leq 2^n$, so the answer to the corresponding instance of ATLEAST-UNSAT is always "yes".

Now consider the counting problem #3SAT, defined as:

INPUT: A 3SAT formula F.
OUTPUT: The number of assignments that satisfy F.

The counting problem #3SAT is known to be complete for the complexity class **#P** (Valiant 1979).

Theorem 2. *$\#3SAT$ is in $\boldsymbol{P}^{\text{BPE(SC-Acyc)}}$.*

Proof. Any instance F of $\#3SAT$ has between 0 and $2^n - 1$ satisfying assignments, where n is the number of variables of the formula. Since ATLEAST-UNSAT reduces to BPE(SC-Acyc), we can perform a binary search over the range $[0, 2^n - 1]$ to find the exact number of non-satisfying assignments for F in linear time. Consequently, we can also find the number of satisfying assignments and it follows that $\#3SAT$ can be solved in linear time given an oracle for BPE(SC-Acyc). □

Corollary 9. *The polynomial time hierarchy is a subset of $\boldsymbol{P}^{\text{BPE(SC-Acyc)}}$.*

Proof. It follows from Theorem 2 and the fact that #3SAT is $\#\mathbf{P}$-complete that $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}^{\#3\text{SAT}} \subseteq \mathbf{P}^{\mathbf{P}^{\text{BPE(SC-Acyc)}}}$. Toda's Theorem (Toda 1991) states that $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$, which gives us $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{P}^{\text{BPE(SC-Acyc)}}} = \mathbf{P}^{\text{BPE(SC-Acyc)}}$. □

Discussion

We have proved that the plan existence problem is **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 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 $\mathbb{C}_n$ denote the directed chain on n vertices. Now, is it the case that $\text{PE}(\{\mathbb{C}_1, \mathbb{C}_2, \ldots\})$ is **PSPACE**-complete? It is known that $\text{PE}(\{\mathbb{C}_1, \mathbb{C}_2, \ldots\})$ 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 **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-**FPT** $\neq$ **W[1]**, exactly characterised the sets of graphs $\mathcal{C}$ such that $\text{PE}(\mathcal{C})$ is in **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 **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 **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 **P** and $\text{PE}(\mathcal{C})$ is not **NP**-hard. Clearly, a characterisation of the **PSPACE**-complete graphs (and also of the X-complete graphs for other complexity classes X within **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 (cf. 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 PE tractable.

Acknowledgments

Anders Jonsson is partially supported by European FP7 project #270019 (SpaceBook). Tomas Lööw is partially supported by the Swedish National Graduate School in Computer Science (CUGS).

References

Bacchus, F., and Yang, Q. 1994. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence* 71:43–100.

Bäckström, C., and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence* 11(4):625–655.

Brafman, R., and Domshlak, C. 2003. Structure and Complexity in Planning with Unary Operators. *Journal of Artificial Intelligence Research* 18:315–349.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69:165–204.

Chen, H., and Giménez, O. 2008. Causal Graphs and Structurally Restricted Planning. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 36–43.

Giménez, O., and Jonsson, A. 2008. The Complexity of Planning Problems with Simple Causal Graphs. *Journal of Artificial Intelligence Research* 31:319–351.

Giménez, O., and Jonsson, A. 2009. Planning over Chain Causal Graphs for Variables with Domains of Size 5 Is NP-Hard. *Journal of Artificial Intelligence Research* 34:675–706.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, 161–170.

Jonsson, P., and Bäckström, C. 1998. Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence* 22(3-4):281–296.

Jonsson, A. 2009. The Role of Macros in Tractable Planning. *Journal of Artificial Intelligence Research* 36:471–511.

Katz, M., and Domshlak, C. 2008. New Islands of Tractability of Cost-Optimal Planning. *Journal of Artificial Intelligence Research* 32:203–288.

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Stockmeyer, L., and Meyer, A. 1973. Word problems requiring exponential time. In *Proceedings of the 5th Symposium on Theory of Computing (STOC)*, 1–9.

Toda, S. 1991. PP is as hard as the polynomial-time hierarchy. *SIAM Journal of Computing* 20:865–877.

Valiant, L. 1979. The complexity of enumeration and reliability problems. *SIAM Journal of Computing* 8(3):410–421.

Williams, B., and Nayak, P. 1997. A reactive planner for a model-based executive. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1178–1185.