Safety Analysis in the NGAC Model

Brian Tan
Brian.Tan@colostate.edu
Colorado State University
Fort Collins, CO, USA

Ewan S. D. Davies
Ewan.Davies@colostate.edu
Colorado State University
Fort Collins, CO, USA

Indrakshi Ray
Indrakshi.Ray@colostate.edu
Colorado State University
Fort Collins, CO, USA

Mahmoud A. Abdelgawad
M.Abdelgawad@colostate.edu
Colorado State University
Fort Collins, CO, USA

Abstract

We study the safety problem for the next-generation access control (NGAC) model. We show that under mild assumptions it is coNP-complete, and under further realistic assumptions we give an algorithm for the safety problem that significantly outperforms naive brute force search. We also show that real-world examples of mutually exclusive attributes lead to nearly worst-case behavior of our algorithm.

CCS Concepts

• **Security and privacy → Access control**; • **Theory of computation → Problems, reductions and completeness**.

Keywords

Access control, Next-generation access control, safety problem, computational complexity

ACM Reference Format:
Brian Tan, Ewan S. D. Davies, Indrakshi Ray, and Mahmoud A. Abdelgawad. 2025. Safety Analysis in the NGAC Model. In *Proceedings of the 30th ACM Symposium on Access Control Models and Technologies (SACMAT '25), July 8–10, 2025, Stony Brook, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3734436.3734444

1 Introduction

Access control policies (ACPs) are an essential technology in cybersecurity. Broadly, given some users and resources, an ACP guarantees the legitimate access of resources by authorized users as well as preventing unauthorized access.

There are several widely-accepted ACP frameworks including Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC). We study a prominent example of ABAC, the Next Generation Access Control (NGAC) model published by the American National Standards Institute [3]. Roughly, the *state* of an NGAC model is equivalent to a directed graph (digraph) whose vertices comprise sets of users, user attributes, resource attributes, resources, and policy classes. The edges of the digraph correspond to relations on the vertices that fall into one of four categories: obligations, prohibitions, associations, and assignments. Given a state, we say that a user u can *access* a resource r if there is a path from u to r in the associated digraph.

An important feature of the NGAC model in applications is the provision for *commands* (also known as *obligations*) that, subject to some conditions, change the state of the model. For example, a command could introduce or delete vertices and edges in the digraph after checking a condition such as the existence of an edge in the current state. The dynamic nature of an NGAC model means that formal analysis is nontrivial, and in this work we are interested in a notion of safety for such models. For us, an NGAC model is *safe* if the execution of any available obligations cannot modify the state in such a way that a new access path from some user u to a resource r is created. The precise definition is somewhat technical, we defer a formal statement to Section 2. The *safety problem* (SP) is the problem of determining, given a description of an NGAC model, whether the model is safe in this sense. An important subtlety is that safety is represented as the *absence* of some concrete structure in the model, namely a sequence of commands whose execution results in a new access path. Thus, it is convenient to consider the complementary problem that could naturally be called the *unsafety problem* or more technically the *co-safety problem* (coSP).

Our main contributions are to give a rigorous definition for the safety problem (and co-safety problem) in the NGAC model, and to show that co-safety is an NP-complete decision problem (in the mono-operational case, see below). We achieve this in two steps, first abstracting the dynamic nature of co-safety in the NGAC model into a problem of path-finding subject to constraints in a static digraph that we call *directed acyclic constrained connectivity* (DACC). We then reduce 3-coloring (3COL) to this decision problem, showing that co-safety is NP-complete via the relation

$$\text{3COL} \le_p \text{DACC} \le_p \text{coSP}$$

where A $\le_p$ B represents the standard notion that there is a polynomial-time reduction from problem A to problem B. That 3COL can be reduced to the co-safety problem in the NGAC model shows that coSP is NP-hard, and proving that coSP $\in$ NP is trivial. While the fact that a problem equivalent to DACC is NP-complete has been discussed in the literature before (cf. Problem GT54 in Garey and Johnson's comprehensive book [10] and the paper [9]), since DACC is not a well-known NP-complete problem we give the short proof for completeness.

Our secondary contributions are to exploit the graph-theoretic nature of the safety problem for the NGAC model to give an algorithm for the safety problem that significantly outperforms naive brute-force search. Though our algorithm is exponential-time in general, it is adaptive to a graph-theoretic property known as the number of *maximal independent sets* (MIS) in a natural "supergraph" constructed from the NGAC model. That is, if the specific NGAC model being studied yields supergraph that has a polynomially-bounded number of MIS then our algorithm decides the safety problem in polynomial time. We also investigate the real-world implications of this result by considering natural structures that occur in real-world NGAC models. Here, our findings are negative and we give a natural class of NGAC models which encode the structure known by the classic combinatorial result typically attributed to Moon and Moser [17, 18] to yield the worst possible case for the number of MIS. These observations link the complexity of safety in the NGAC model to a large body of work in extremal combinatorics and the algorithmic study of enumerating MIS. Writing $\mu(C)$ for the set of MIS in a simple graph C, we know [17, 18] that $|\mu(C)| \le 1.45^{|V(C)|}$, and this forms the basis of the worst-case time complexity of our algorithm. Naive search for (maximal) independent sets in C takes much longer: time $\Omega(2^{|V(C)|})$.

1.1 Related work

The kind of safety problem we study here has a long and rich history in cybersecurity. The foundational work of Harrison, Ruzzo and Ullman [11] defines an ACP framework whose state consists of sets of *rights* in the entries of a matrix indexed by subject-object pairs. We refer to this widely studied model, which appears in many introductory texts [4, 8, 20], as the HRU model. As with the more modern NGAC model, the HRU model is dynamic and features state-changing commands.

In part due to the flexibility of the HRU model, the safety problem for the model is rather subtle. Harrison, Ruzzo and Ullman themselves demonstrated [11] that different natural formulations of the safety problem for the HRU model can be NP-complete or even undecidable. One of the restrictions that permits the NP-completeness proof is that commands are mono-operational (which we explain later), and we make an analogous assumption in our main result too.

Aside from the fact that natural variations on safety can have such wildly different computational complexities, the challenges of safety analysis are further evidenced by the comprehensive review the HRU model due to Tripunitara and Li [26]. Tripunitara and Li refined the definitions of safety problems for the HRU model from the original work to resolve several ambiguities. They also corrected several errors in widely-circulated proofs of complexity results on safety problems for the HRU model. Their rigorous treatment of the computational complexity of safety for a concrete ACP framework serves as direct inspiration for our work. In the wake of the original study of Harrison, Ruzzo and Ullman, there are works defining a number of alternative ACP frameworks and a broad literature on safety [2, 5, 12, 13, 15, 19, 22, 23, 24, 25].

1.2 Organization

In Section 2 we state precisely what we mean by the NGAC model and define the safety problem that we study. In Section 3 we give a precise definition of directed acyclic constrained connectivity (DACC) in order to strip away features of the NGAC model which are not especially important from the perspective of the complexity of the safety problem. We then give the reduction from 3-coloring to DACC and the reduction from DACC to the co-safety problem. In Section 4 we give our algorithm for the safety problem that exploits maximal independent sets, as well as proving correctness and running time bounds. Finally, in Section 5 we discuss aspects of the performance of our algorithm in a real-world context.

2 Preliminaries

An NGAC model M consists of an 11-tuple

$$M = (U, UA, R, RA, R_\psi, A_U, A_R, ASC, P, V, COM)$$

where

- U is a set of users,
- UA is a set of user attributes,
- R is a set of resources,
- RA is a set of resource attributes,
- R_ψ is a set of access rights,
- A_U is a set of assignment edges,
- A_R is a set of assignment edges
- ASC is a set of (labeled) association edges,
- P is a set of (labeled) prohibition edges,
- V is a "universe" set of entities that can be in the model,
- COM is a set of commands.

We require that the sets U, UA, R, RA and R_ψ are pairwise-disjoint and $U \cup UA \cup R \cup RA \subseteq V$[1]. The set $U \cup UA \cup R \cup RA$ forms the vertex set of the digraph G representing the state of the model. The access rights R_ψ are used as labels for edges in ASC and P, and the sets A_U, A_R, ASC and P form edges of G. We define G piece-by-piece as follows.

The sets A_U and A_R correspond to unlabeled, directed edges in G on the vertex sets $U \cup UA$ and $R \cup RA$ such that

$$A_U \subseteq (U \times UA) \cup (UA \times UA)$$
$$A_R \subseteq (RA \times R) \cup (RA \times RA).$$

It is standard (e.g. [7]) to assume that the subgraphs of G given by $(U \cup UA, A_U)$ and $(R \cup RA, A_R)$ are acyclic and to refer to them as the *user DAG* and the *resource DAG* respectively. The set ASC is a set of labeled edges satisfying $ASC \subseteq UA \times RA \times R_\psi$, where we think of $a = (ua, rsa, r)$ as an edge in G from the user attribute ua to the resource attribute rsa labeled with the access right r. The prohibition edges P satisfy $P \subseteq UA \times RA \times R_\psi$ and are also thought of as labeled edges of G in the same way.

The universe set V represents the collection of all possible users, user attributes, resources, and resource attributes that may be added to the state of the model. That is, we can think of V as a "reservoir" of vertices that do not yet exist in G but that may be added by commands. We restrict R_ψ and V to be finite.

[1]Alternative formulations are possible, e.g. $R \subset RA$ is assumed in [6], and the universe and commands are elided.

To reflect the dynamic nature of the NGAC model, wherein commands can change the state digraph G, it is convenient to index the state graphs with a time variable t. We call the initial state digraph G_0 and the execution of a command c at time $t \geq 1$ takes G_{t-1} as input and produces another state digraph G_t. The set COM is a set of commands of the following form.

```
command $\alpha(X_1, X_2, \ldots, X_k)$
    if $cond_1$ and $cond_2$ and $\cdots$ and $cond_m$ then
        $op_1$
        $op_2$
        $\ldots$
        $op_n$
```

where $X_1, \ldots, X_k$ are the formal parameters of the command, $cond_i$ is a condition and op_j is one of the primitive operations detailed in Table 1 (whose input must be one of the formal parameters of α). We require that each condition is of the form $s \notin G_{t-1}$ where G_{t-1} is the state digraph before the execution of the command and s is either an entity in V or a possible edge in a state digraph[2]. Each primitive operation consists of a single addition or deletion of an element of the sets U, UA, R, RA, A_U, A_R, ASC, and P. Thus, there are 16 primitive operations. Note that by convention, for each of the primitive operations that destroys a vertex v of G (i.e. an element of $U \cup UA \cup R \cup RA$) also destroys all edges of G that involve v. For brevity, this convention is suppressed from the notation in Table 1.

Another standard assumption that we make is that all commands are *mono-operational*, which means each command contains one primitive operation, i.e. $n = 1$ in the above description of a command. Since all primitive operations only take one input, then we can also say that we only need one formal parameter per command, i.e. $k = 1$. Thus, we can drop the subscripts on the formal parameter and operation in a command. We can also assume that m is finite since the fact that V is finite implies that the number of possible conditions is also finite. Note that for each command, the set of possible values for the formal parameter is finite, since every set mentioned in the conditions is finite too.

In our algorithm (Section 4), we require a few additional, mild assumptions.

(1) Only commands $\alpha(X)$ whose single operation is of the form "**create** ... X", where X is an edge of the state graph (i.e. a user assignment, resource assignment, association or prohibition), have any condition.
(2) The conditions in such a command are of the form $e_1 = X$ to make sure the command can only be executed for the input e_1, or $e_2 \notin G_{t-1}$ where e_2 is a potential edge of the state digraph and G_{t-1} is the state digraph before the execution of the command. These conditions can prevent the creation of a specific edge due to the presence of existing edges.
(3) All vertices and edges which may be present in the state digraph have unconditional destroy commands.

It is natural to consider conditions only on the creation of edges of the state graph as it is edges which determine access to resources and so guarding their creation with conditions is flexible enough to model real-world access policies. Since creating a node of the state graph with no adjacent edges cannot create undesirable access paths, it is not a significant restriction to demand that vertex creation is unconditional.

On the other hand, edge creation conditions allow us to model natural constraints, for example that a user must have the attribute "student" in order to be allowed to gain the attribute "teaching assistant". This natural type of constraint is expressible within the restrictions we give in assumption 2.

Assumption 3 is also important in real-world access control models, as e.g. when a user leaves an organization one should delete all vertices in the state graph corresponding to that user. Though deleting vertices corresponding to attributes is not normally required, allowing for this does not significantly alter the NGAC model or its applicability. The situation is similar with edges. Promotion or other role changes of personnel in an organization may require the deletion of edges in the state graph (e.g. the promotion of a user to a higher rank with greater security privileges). While association or prohibition edges are normally static, they may need to be deleted if the organization suddenly adopts stricter access policies.

Despite these further restriction on the conditions in commands, NGAC models satisfying our assumptions are sufficiently expressive to allow for *mutually exclusive attributes*. These are groups of attributes such that a user or resource can have at most one of the attributes in the group. Natural examples of mutually exclusive attributes abound. In the context of separation of duty (which is commonly discussed in the context of RBAC [14]) one might choose a "role" to be a group of mutually exclusive user attributes. An example due to Sandhu and Samarati [21] is that the "authorizer" and "preparer" roles in the context of paychecks should be mutually exclusive. The system is more vulnerable to abuse if a single user has the right to both prepare and authorize checks. If we separate those duties then coordination is required for abuse to occur. More generally, in ABAC there are natural attributes that users cannot hold simultaneously. For example in the category of "age" the user attributes "minor" and "adult" are mutually exclusive. Resource attributes can also be mutually exclusive, for example a "security clearance" could be exactly one of "top secret", "secret", "confidential" or "public".

Since assigning users or resources roles takes the form of creating edges from U or R to UA or RA in the state digraph, we can implement mutually exclusive attributes in the presence of the assumptions 1–3 with create commands whose conditions check that the user or resource which is to gain an attribute does not already have an attribute in the mutually exclusive group. We will see in Section 5 that natural groups of mutually exclusive attributes are a significant obstacle for the algorithm we propose in Section 4.

2.1 The safety problem

The following definition simplifies the study of safety in the NGAC model.

Definition 2.1. Given a state digraph, we define for each $r \in R_{\psi}$ the *access relation* to be binary relation $\rightarrow^r$ (formally, a subset of $U \times R$) such that $u \rightarrow^r rs$ if and only if there exists a path in the state digraph from u to rs that goes through an association edge labeled with r.

[2] i.e. a suitable element of $V \times V$ or $V \times V \times R_{\psi}$.

Table 1: List of Primitive Operations

Operation	Conditions	Action
create user u	$u \notin U \wedge u \in V$	$U \mapsto U \cup \{u\}$
create user attr. ua	$ua \notin UA \wedge ua \in V$	$UA \mapsto UA \cup \{ua\}$
create res. rs	$rs \notin R \wedge rs \in V$	$R \mapsto R \cup \{rs\}$
create res. attr. rsa	$rsa \notin RA \wedge rsa \in V$	$RA \mapsto RA \cup \{rsa\}$
create user assign. au	$au \notin A_U \wedge au \in (U \times UA) \cup (UA \times UA)$	$A_U \mapsto A_U \cup \{au\}$
create res. assign. ar	$ar \notin A_R \wedge ar \in (RA \times R) \cup (RA \times RA)$	$A_R \mapsto A_R \cup \{ar\}$
create assoc. a	$a \notin ASC \wedge a \in UA \times RA \times R_\psi$	$ASC \mapsto ASC \cup \{a\}$
create prohib. p	$p \notin P \wedge p \in UA \times RA \times R_\psi$	$P \mapsto P \cup \{p\}$
destroy user u	$u \in U$	$U \mapsto U \setminus \{u\}$
destroy user attr. ua	$ua \in UA$	$UA \mapsto UA \setminus \{ua\}$
destroy res. rs	$rs \in R$	$R \mapsto R \setminus \{rs\}$
destroy res. attr. rsa	$rsa \in RA$	$RA \mapsto RA \setminus \{rsa\}$
destroy user assign. au	$au \in A_U$	$A_U \mapsto A_U \setminus \{au\}$
destroy res. assign. ar	$ar \in A_R$	$A_R \mapsto A_R \setminus \{ar\}$
destroy assoc. a	$a \in ASC$	$ASC \mapsto ASC \setminus \{a\}$
destroy prohib. p	$p \in P$	$P \mapsto P \setminus \{p\}$

When studying an NGAC model the state graph is dynamic and hence for each $r \in R_\psi$, the relation $\to^r$ is also dynamic. Given a sequence S of commands, we write $u \to^r_S rs$ to mean that starting from the current state digraph, after applying the sequence of commands S we have $u \to^r rs$ in the resulting state digraph.

Note that there are no primitive operations that modify R_ψ and so we cannot create or destroy access relations over time, only modify them. We can now define the concept of safety that we study.

Definition 2.2. Given an NGAC model M, we say that M is *safe* if, for all rights $r \in R_\psi$, and all finite sequences of commands S whose elements are in COM, we have that

$$\to^r_S \;\subseteq\; \to^r .$$

The subset notation above means that after the execution of S, there cannot be a new element of any access relation. That is, no matter what sequence of commands is performed on the model goes through, no new access is gained.

Definition 2.3. The *safety problem* SP is the decision problem that, given an input NGAC model M, returns "Yes" if the model is safe in the sense of Definition 2.2.

The *co-safety problem* coSP is the complement of SP. That is, coSP is a decision problem and the answer to coSP(M) is "Yes" if and only if the answer to SP(M) is "No".

Note that throughout, we assume that the NGAC model input to these safety problems satisfies the assumptions outlined above. In particular, V is finite, and the commands in COM are mono-operational with the structure required above.

3 Computational Complexity

In this section we prove that the co-safety problem is NP-complete. We will first define a graph-theoretic abstraction of co-safety that we call *directed acyclic constrained connectivity* (DACC). This problem serves as an intermediate step is our reduction that "smooths out" several features of the NGAC model that are not critical to the complexity of the safety problem. Recall that we show

$$\text{3COL} \leq_p \text{DACC} \leq_p \text{coSP},$$

where 3COL is the classic 3-coloring decision problem (see e.g. [10, GT4]).

Definition 3.1. Given a directed acyclic graph $\Gamma = (V, E)$, a *constraint graph* for Γ is a graph $C = (E(\Gamma), E')$ where $e_1e_2 \in E'$ represents the constraint that edges e_1 and e_2 cannot both exist.

Given a constraint graph C for Γ, we say that a subgraph $\Gamma' \subseteq \Gamma$ is *valid* if the edges of Γ' form an independent set in C. That is, at most one edge of Γ from each constraint can be present in Γ'

The DACC problem is about the existence of paths in valid subgraphs of a directed acyclic graph G.

Definition 3.2. The *directed acyclic constrained connectivity* problem (DACC) is the decision problem which, given as input a directed acyclic graph $\Gamma = (V, E)$, a constraint graph C, and a pair of vertices $s, t \in V$, asks whether there exists a valid subgraph of Γ in which there is a (directed) path from s to t.

One can view DACC as a modification of the classic st-connectivity problem where we introduce constraints on the path from s to t that one should find: we restrict our attention to paths that are valid subgraphs of Γ given some constraint graph C. Our first result is that DACC is NP-complete.

Theorem 3.3. DACC *is* NP-*complete.*

Proof. It is easy to see that DACC is in NP as one can take the certificate of a "Yes" answer to be the st-path itself, and one can easily check validity in polynomial time by looping through the constraints.

It remains to show that DACC is NP-hard, which we do by giving a reduction 3COL $\leq_p$ DACC.

Let G be an input to 3COL, and let $n = |V(G)|$. We construct an input (Γ, C, s, t) for DACC such that G is 3-colorable if and only if

(Γ, C, s, t) yields the answer "Yes" in DACC. A key property will be that st-paths in Γ correspond to 3-colorings (not necessarily proper) of the vertices of G.

First, let $V(G) = \{v_1, \ldots, v_n\}$ under some arbitrary ordering. To construct Γ we include a copy of each v_i in $V(G)$ and, for each i, three vertices labeled R_i, G_i, and B_i. Then, we create (directed) edges from each v_i to each of $\{R_i, G_i, B_i\}$, and for $i < n$ from each $\{R_i, G_i, B_i\}$ to v_{i+1}. Finally, we create new vertices s and t, connect s to v_1 and each of $\{R_n, G_n, B_n\}$ to t. This process takes time $O(n)$.

The DAG Γ is built in such a way that an edge from v_i to R_i, G_i or B_i can be used to represent an assignment of v_i to the respective colors red, green, and blue. We add constraints to C such that for each i

1. at most one of the edges v_iR_i, v_iG_i, and v_iB_i can exist.
2. For adjacent vertices v_iv_j, at most one of the edges v_iR_i and v_jR_j exist, and similar for the colors G and B.

The vertex set of C is simply the edge set of Γ. In time $O(n)$ we can add the "single assignment constraints" described in 1. Note that this comprises the formation of triangles on the triples of vertices $\{v_iR_i, v_iG_i, v_iB_i\}$ for each i. Similarly, in time $O(n^2)$ we can add the "coloring constraints" described in 2.

Now that we have our input (Γ, C, s, t) constructed in polynomial time, we must show that "Yes" for G in 3COL corresponds to "Yes" for (Γ, C, s, t) for DACC.

Let G be 3-colorable, and let $\varphi : V(G) \to \{R, G, B\}$ a proper coloring of G. We show that the path

$$P = (s, v_1, \varphi(v_1), v_2, \varphi(v_2), \ldots, v_n, \varphi(v_n), t)$$

is valid in Γ. By construction P cannot violate any of the single assignment constraints, and since φ is proper it does not violate any of the coloring constraints.

Suppose that G has no proper 3-coloring. Then every path P from s to t in Γ is invalid, because each such path corresponds to a function $\varphi : V(G) \to \{R, G, B\}$ in the natural way. Since G is not 3-colorable there must exist a pair v_iv_j of adjacent vertices which get the same color under φ, but this means that the edges $v_i\varphi(v_i)$ and $v_j\varphi(v_j)$ are both in the path P. But this violates one of the coloring constraints. □

Turning to the safety problem, we show coNP-completeness in two steps.

Lemma 3.4. *coSP ∈ NP.*

Proof. If an NGAC model M (with the standard 11-tuple notation) is a "Yes" instance of coSP, then there is a tuple $(u, rs, r) \in U \times R \times R_\psi$ and a sequence of commands S such that $u \not\twoheadrightarrow^r rs$ and $u \in\to^r_S rs$. We let the certificate be a combination of the tuple and the sequence S. To verify the certificate, we need to check both the conditions.

To check that $u \not\twoheadrightarrow^r rs$ we start with the state digraph G_0 and remove all edges in ASC that are not labeled r, then use breadth-first search to check that there is no path from u to rs. This can be done in time polynomial in the size of the input M to the coSP problem.

To check that $u \in\to^r_S rs$, we start with G_0 and apply the sequence S of commands. Since each command and possible input it can take are part of the input M, this takes time polynomial in the size of M. Suppose that this gives the state digraph G_t. Then, we can remove all edges in the the ASC part of G_t that are not labeled r, and use breadth-first search to check that there is a path from u to rs in this graph. Again, this takes polynomial time. □

We now give a reduction showing hardness. The key idea is that we can build an NGAC model M in which commands add or remove edges of the state digraph in such a way that the constraints in an instance (Γ, C, s, t) of DACC are respected. Figure 1 shows the basic idea, where we embed valid subgraphs of Γ in the subgraph of the state digraph induced by the user attributes UA. That is, we embed a hard instance (Γ, C, s, t) of DACC in the subgraph of the state digraph of the NGAC model induced by the user attributes UA. It suffices construct an NGAC model with a single user u connected by an edge to the user attribute represented by the source s in Γ, and a single edge from the target t of Γ to a single resource attribute rsa (labeled by a single right r). Finally, we have a single resource rs to which rsa is connected by an edge. The rest of the proof consists of showing that with a suitably-defined set of commands COM, the states of the model we can explore by running commands correspond to valid subgraphs of Γ.

Theorem 3.5. *coSP is NP-complete.*

Proof. Sine we have Lemma 3.4 and Theorem 3.3, it suffices to show that DACC $\leq_p$ coSP.

Let (Γ, C, s, t) be the input of DACC. We construct an NGAC model M with the standard 11-tuple notation as follows. Let $U = \{u\}$, $UA = V(\Gamma)$, $R = \{rs\}$, $RA = \{rsa\}$, $A_U = \{(u, s)\}$, $A_R = \{(rsa, rs)\}$, $ASC = \{(t, rsa, r)\}$, $P = \emptyset$, $R_\psi = \{r\}$, $V = \{u, rs, rsa\} \cup V(\Gamma)$, and let COM be the set of commands defined as follows.

First, in COM we include an unconditional destroy command for every vertex and and edge of Γ. Then, for every constraint $e_1e_2 \in E(C)$, we add to C the two commands

```
command $\alpha_{1,2}(X)$
    if $e_1 == X$ and $e_2 \notin G_{t-1}$ then
        create user assign. $X$
command $\alpha_{2,1}(X)$
    if $e_2 == X$ and $e_1 \notin G_{t-1}$ then
        create user assign. $X$
```

The conditions in the command ensure that when starting from a state in which at most one of e_1 and e_2 exist at the same time, then this remains true.

This gives an input M to coSP, one that can clearly be constructed in polynomial time in the size of Γ.

Given the command set COM, it is straightforward to see that the valid subgraphs of Γ are in bijection with possible states of the user DAG, the subgraph of the state digraph induced by the user attributes UA. Moreover, we have $u \not\twoheadrightarrow^r rs$ in the initial state graph G_0 by construction. It remains to show that the answer for (Γ, C, s, t) in DACC is "Yes" if and only if the answer for M in coSP is "Yes".

Let (Γ, C, s, t) be a "Yes" instance of DACC. Then there is a valid subgraph Γ' of Γ in which there is a path from s to t. But this means that there is a sequence of commands S that results in the state digraph G_t having a path from u to rs through an edge in ASC labeled R. That is, we have $u \to^r_S rs$ and hence M gives a "Yes" in coSP.

Let (Γ, C, s, t) be a "No" instance of DACC. Then, there is no valid subgraph of Γ that connects s and t. But the conditions on edge creation commands in COM mean that the only valid states of M are digraphs in which the subgraph induced by UA is a valid subgraph of Γ. Thus, there is never an st-path in a state digraph of M and hence there is never an access relation of the form $u \rightarrow^{r}_{S} rs$ for any sequence S of commands. Then M is a "No" instance of coSP as required. □

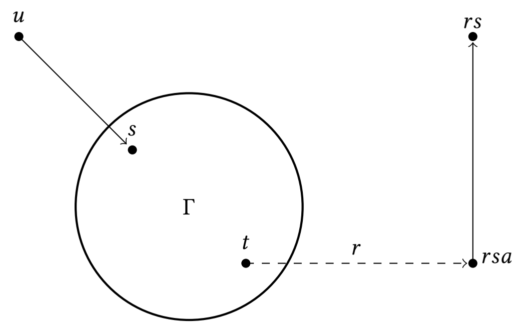

Figure 1: Reduction from DACC **to** coSP.

4 Algorithm

In this section, we will introduce an algorithm that solves the safety problem for the NGAC model significantly faster than naive brute-force search. We will first give an algorithm to solve DACC, and then explain how we can transform a safety problem input to an input for DACC to use the algorithm.

4.1 Solving DACC

Let (G, C, s, t) be an input to DACC. Since we are trying to find a path in a valid subgraph of G, it suffices to enumerate *maximal* valid subgraphs of G. This is because adding edges to a valid subgraph cannot destroy the path we are looking for. In terms of the constraint graph C, this means it suffices to enumerate the maximal independent sets $\mu(C)$ of C. An *output sensitive* enumeration algorithm for a set S with delay d takes time at most $d|S|$ to enumerate S. We think of the delay as an upper bound on the computation time needed per element for the numeration. Our algorithm simply uses a state-of-the-art output-sensitive enumeration algorithm for MIS [16][3] and checks for the existence of an st-path in each of the returned MIS of C (which corresponds to a valid subgraph of G) with breadth-first search. See Algorithm 1. The algorithm we use [16] for enumerating MIS has a time delay of $M(n)$ when the input is an n-vertex graph, where $M(n)$ is the time needed to multiply two $n \times n$ matrices. The best available result [1] shows that $M(n) = O(n^{2.372})$ (though exponents approximately 2.37 have been known for some time).

[3]The algorithm AllMaxCliques in particular, though MIS in G are maximal cliques in the complement of G so we have to take the complement first.

Algorithm 1 An algorithm for DACC based on enumeration of MIS.

```
1: procedure DACC-via-MIS($G = (V, E), C, s, t$)
2:     for $I \in \mu(C)$ do                      ▷ Use the algorithm of [16]
3:         $G' \leftarrow (V, I)$
4:         if $\exists$ $st$-path in $G'$ then      ▷ Use breadth-first search
5:             return True
6:     return False
```

A well-known result in extremal combinatorics [17, 18] shows that the number of MIS in an n-vertex graph is at most $3^{n/3} < 1.45^{n}$. The result is often attributed to Moon and Moser [18], but was proved independently by Miller and Muller [17] several years before. Simpler proofs were given more recently by Vatter [27] and Wood [28]. This result forms the basis of our analysis of the worst-case time complexity of our algorithm for DACC. The upper bound is achieved by a disjoint union of triangles, which we refer to later in our consideration of whether real-world examples are close to the worst case of this algorithm. Overall, in the worst case this enumeration of MIS significantly outperforms a naive search over all sets (filtering out the non-independent ones), and more importantly there are classes of graphs with a polynomially-bounded number of maximal independent sets and thus the algorithm takes polynomial time on such graphs. We bound the running time of Algorithm 1 below.

Lemma 4.1. *Let (G, C, s, t) be an input to Algorithm 1 such that $n = |V(G)|$, $m = |E(G)|$ and $\mu(C)$ is the set of MIS in C. Then the running time is*

$$O(n + m) \cdot M(m) \cdot \mu(C) \leq O(n^{6.75}) \cdot \mu(C) \leq O(1.45^{n^2}).$$

Proof. We have an input (G, C, s, t) for the DACC problem with $n = |V(G)|$ and $m = |E(G)|$. Algorithm 1 is simple and loops over the MIS of C, executing for each MIS a breadth-first search in a sungraph of G. This search takes time $O(n + m)$ (the worst-case time complexity of standard breadth-first search). Since $|V(C)| = |E(G)| = m$, the algorithm [16] takes time $M(m) \cdot \mu(C)$ to do the required enumeration. This is because $M(m)$ is the time delay and $\mu(C)$ is the size of the set being enumerated. Note that $m \leq n^2$ and $M(m) \leq O(m^{2.372})$ to see the first inequality in the running time bound. For the final inequality note that $\mu(C) \leq 3^{m/3} < 3^{n^2/3} < 1.45^{n^2}$. □

4.2 Solving the safety problem

Let $M = (U, UA, R, RA, R_{\psi}, A_U, A_R, ASC, P, V, COM)$ be an input to the safety problem SP and that M satisfies our assumptions. Recall that in addition to having mono-operational commands, we require the enumerated assumptions 1–3 stated in Section 2.

Let N be the length of the input M in a natural binary encoding. We will construct an input (Γ, C, s, t) for DACC in time polynomial in N, and use algorithm 1 to solve DACC for (Γ, C, s, t) in order to solve the safety problem for M.

4.2.1 Constructing Γ. We let Γ be the *supergraph* of M, meaning that Γ is the result of starting with the initial state digraph of M and executing, for all valid inputs, all possible commands in COM whose primitive operation is one of the form "**create** …", *without* checking the condition(s) in the command. Since we assume that COM is finite, and V and hence the vertex set of Γ is also finite, and we know that the number of edges of Γ is bounded by polynomial in $|V|$ and $|R_\psi|$, it takes time polynomial in the size of the input M (which includes COM, R_ψ and V) to construct Γ. We also have the property that $G_t \subseteq \Gamma$ where G_t is the state digraph at any given time t.

4.2.2 *Constructing C.* We construct a constraint graph C for the supergraph Γ which enforces the conditions of the create commands in the model M. The goal is to do this in such a way that valid subgraphs of Γ correspond to possible state digraphs G_t of the model M.

We initialize an empty constraint graph $C = (E(\Gamma), \emptyset)$. By assumptions 1 and 2, each edge creation command has one parameter X corresponding to an edge of Γ and must have a conditional of the form

$$e == X \textbf{ and } cond_1 \textbf{ and } cond_2 \textbf{ and} \cdots \textbf{and } cond_m,$$

where each $cond_i$ is of the form $e_i \notin G_{t-1}$ for some potential edge (element of $V \times V$ or $V \times V \times R_\psi$). We may assume that $e_i \in \Gamma$, else the command can never execute and we may remove it. Then for each such command c with this structure we add the edges

$$\{ee_i : i \in M\}$$

to C. This will make sure that in a valid subgraph $\Gamma' \subseteq \Gamma$, if e exists in Γ' then the edges in the conditions of c must not exist.

Lemma 4.2. *The valid subgraphs Γ' of Γ correspond to possible states of the model M.*

Proof. Let G_0 be the initial state digraph of M. Writing

$$M = (U, UA, R, RA, R_\psi, A_U, A_R, ASC, P, V, COM),$$

we have

$$G_0 = (U \cup UA \cup R \cup RA, A_U \cup A_R \cup ASC \cup P).$$

We first prove that any valid subgraph Γ' of Γ is a valid state of the model. Given Γ', we want to sequence of commands S which, starting from G_0, puts the model in state Γ'.

First, for all vertices and edges that are in G_0 but not in Γ', we have to add a command to S to destroy these elements of the state digraph. Note that by assumption 3, all vertices and edges must have an unconditional destroy command, thus we are able to destroy the necessary vertices and edges.

Next, we have to create all the vertices and edges that are in Γ' but not in G_0. Since Γ was constructed to be the supergraph of M, for every necessary vertex or edge creation there is a command to create the required structure in the model. If this command is unconditional then we simply add it to the sequence S. By assumption 1, only edge creation commands have conditions, and by assumption 2 these conditions merely check for the absence of some edges. But by the construction of C and the fact that Γ' is valid, the conditions of any create command that we need to add to S will be true at the time of execution.

Now we prove that any state of the model is a valid subgraph of Γ. Let S be a sequence of commands and suppose that running S from G_0 yields the state digraph G_t.

By the construction of the command set COM, it is easy to prove (formally, by induction on the length of the sequence of commands S) that G_t satisfies the constraints encoded by C. The key to the proof is the construction of the constraints in C from the commands themselves. □

4.2.3 *Testing for access paths.* Now that we have constructed Γ and C, we want to loop over all possible access paths and check that, for each path that does not exist in the G_0, there is no sequence of commands S that results in the access path existing.

First, we fix a tuple $(u, rs, r) \in U \times R \times R_\psi$. Then, if $u \not\to^r rs$ we run let Γ^r be the subgraph of Γ obtained from Γ by deleting all association edges not labeled with r and run Algorithm 1 with input (Γ^r, C, u, rs). If the answer is "Yes", then there must exist a sequence of commands S such that $u \to^r_S rs$, which means we can return "No" for the safety problem with input M.

If all queries DACC(Γ^r, C, u, rs) in the loop above tuple return "No", then M is safe and we return "Yes" in the safety problem.

4.3 Analysis of running time

Constructing Γ takes time at most

$$O(|COM| \cdot |V|^2 \cdot |R_\psi|)$$

as the bulk of the construction occurs in a loop over each possible labeled edge of state the digraph in the model, where we check for a create command for that edge and add it to Γ.

Constructing C takes time at most

$$O(|COM| \cdot m_{\max} \cdot |O|^2 \cdot |R_\psi|),$$

where $m_{\max} \leq O(|V|^2|R_\psi|)$ is the maximum number of conditions in any command in COM

Testing for access paths takes time at most

$$|O|^2|R_\psi| \cdot O(|V|^2) \cdot M(|V|^2) \cdot \mu(C)$$

because we execute for each tuple $(u, rs, r) \in V \times V \times R_\psi$ Algorithm 1 with the input (Γ^r, C, u, rs). This Γ^r has at most $|V|$ vertices and $|V|^2$ edges, and hence $\mu(C) \leq 1.45^{|V|^2}$

Therefore the total runtime is at most

$$\text{poly}(|M|) \cdot 1.45^{|V|^2},$$

where $|M|$ is the size of the input M and $|V|$ is the size of the set V of possible objects.

5 Real-world examples and MIS

In this section we observe that real-world NGAC models can elicit nearly worst-case running time of the algorithm to solve the safety problem described in Section 4. The main contributor to the running time is the number $\mu(C)$ of maximal independent sets in the constraint graph C that is built from the commands of the model which enforce separation of duty.

In practice, we often have small groups of mutually exclusive attributes. For example, the attributes *teacher*, *student*, and *staff* might be (pairwise) mutually exclusive in an NGAC model designed for access control in an educational setting. To enforce this fact in the model we would have edge creation commands that will only create e.g. the edge from a user u to the user attribute *teacher* if neither the edges from u to *student* nor from u to *staff* are present. This leads to a triangle in C on the vertices corresponding to the three mutually exclusive attributes. More generally, a set of k mutually exclusive attributes leads to a clique of size k in C. If there are no other attributes that are incompatible with the set of mutually exclusive ones then this clique will be separated from the rest of the graph. So, for each set K of mutually exclusive and otherwise non-interacting attributes, we see a clique in C disjoint and unconnected to the rest of the graph.

But a disjoint union of small cliques leads to nearly the worst case for the number of MIS in a graph. Indeed, the results discussed in the introduction [17, 18] show that a disjoint union of *triangles* is actually the worst case. As seen from the above example, a common organizational structure leads to the creation of a small clique in C, so we can expect that typical NGAC models present somewhat challenging inputs for our algorithm to handle.

Graphs of very high minimum degree have a polynomial number of MIS, but we do not expect typical organizational structures to yield this kind of constraint graph. This would require that all attributes are only compatible with a constant-sized set of other attributes. In the presence of user and resource attributes this seems very unlikely.

6 Conclusions and future work

In this paper, we analyzed the computational complexity of the Safety Problem in NGAC models. We show that under mild assumptions, the Safety Problem in NGAC models is coNP-Complete. We also propose an algorithm to solve the Safety Problem that exploits the combinatorial nature of this problem which outperforms naive brute-force algorithms. We show that mutually exclusive attributes drive the computational complexity of the problem and the running time of our algorithm.

In future work, it would be interesting to test our algorithm with real-world or synthetic data to see if the safety problem tends to be intractable on realistic instances. It would also be interesting to study the fine-grained aspects of complexity subject to parameters that control the number of groups of mutually exclusive attributes and their size.

Acknowledgments

The authors would like to thank the anonymous referees for their comments and suggestions.

This work is supported by the U.S. National Science Foundation under Grant No. CCF-2309707 and the U.S. National Institute of Standards and Technology under Grant No. 60NANB23D152.

References

[1] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. 2025. More Asymmetry Yields Faster Matrix Multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Proceedings. Society for Industrial and Applied Mathematics, 2005–2039. doi:10.1137/1.9781611978322.63.

[2] P.E. Ammann and R.S. Sandhu. 1991. Safety analysis for the extended schematic protection model. In *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Comput. Soc. Press, Oakland, CA, USA, 87–97. ISBN: 978-0-8186-2168-0. doi:10.1109/RISP.1991.130777.

[3] ANSI INCITS. 2020. *Information Technology - Next Generation Access Control (NGAC)*. ANSI.

[4] Matt Bishop. 2003. *Computer Security: Art and Science*. Addison-Wesley Professional. ISBN: 978-0-201-44099-7.

[5] Timothy A. Budd. 1983. Safety in grammatical protection systems. *International Journal of Computer & Information Sciences*, 12, 6, 413–431. doi:10.1007/BF00977968.

[6] Erzhuo Chen, Vladislav Dubrovenski, and Dianxiang Xu. 2023. Coverage-Based Testing of Obligations in NGAC Systems. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies* (SACMAT '23). Association for Computing Machinery, New York, NY, USA, 169–179. doi:10.1145/3589608.3593832.

[7] Erzhuo Chen, Vladislav Dubrovenski, and Dianxiang Xu. 2021. Mutation Analysis of NGAC Policies. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. ACM, Virtual Event Spain, 71–82. ISBN: 978-1-4503-8365-3. doi:10.1145/3450569.3463563.

[8] Dorothy Elizabeth Robling Denning. 1982. *Cryptography and Data Security*. Addison-Wesley, Reading, Mass. ISBN: 978-0-201-10150-8.

[9] H.N. Gabow, S.N. Maheshwari, and L.J. Osterweil. 1976. On Two Problems in the Generation of Program Test Paths. *IEEE Transactions on Software Engineering*, SE-2, 3, 227–231. doi:10.1109/TSE.1976.233819.

[10] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness. A Series of Books in the Mathematical Sciences.* W. H. Freeman, New York. ISBN: 978-0-7167-1045-5.

[11] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. 1976. Protection in operating systems. *Commun. ACM*, 19, 8, 461–471. doi:10.1145/360303.360333.

[12] A. K. Jones, R. J. Lipton, and L. Snyder. 1976. A Linear time algorithm for deciding security. In *17th Annual Symposium on Foundations of Computer Science (SFCS 1976)*. IEEE, Houston, TX, USA, 33–41. doi:10.1109/SFCS.1976.1.

[13] Ninghui Li and Mahesh V. Tripunitara. 2006. Security analysis in role-based access control. *ACM Transactions on Information and System Security*, 9, 4, 391–420. doi:10.1145/1187441.1187442.

[14] Ninghui Li, Mahesh V. Tripunitara, and Ziad Bizri. 2007. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.*, 10, 2, 5–es. doi:10.1145/1237500.1237501.

[15] R. J. Lipton and L. Snyder. 1977. A Linear Time Algorithm for Deciding Subject Security. *Journal of the ACM*, 24, 3, 455–464. doi:10.1145/322017.322025.

[16] Kazuhisa Makino and Takeaki Uno. 2004. New Algorithms for Enumerating All Maximal Cliques. In *Algorithm Theory - SWAT 2004*. Torben Hagerup and Jyrki Katajainen, (Eds.) Springer, Berlin, Heidelberg, 260–272. ISBN: 978-3-540-27810-8. doi:10.1007/978-3-540-27810-8_23.

[17] Raymond E Miller and David E Muller. 1960. A Problem of Maximum Consistent Subsets. Tech. rep. RC-240. IBM Research, JT Watson Research Center, Yorktown Heights, NY.

[18] J. W. Moon and L. Moser. 1965. On cliques in graphs. *Israel Journal of Mathematics*, 3, 1, 23–28. doi:10.1007/BF02760473.

[19] Rajeev Motwani, Rina Panigrahy, Vijay Saraswat, and Suresh Ventkatasubramanian. 2000. On the decidability of accessibility problems (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*. ACM, Portland Oregon USA, (May 2000), 306–315. ISBN: 978-1-58113-184-0. doi:10.1145/335305.335341.

[20] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Lizzie Coles-Kemp. 2024. *Security in Computing*. (Sixth edition ed.). Addison-Wesley, Boston Amsterdam London. ISBN: 978-0-13-789121-4.

[21] R.S. Sandhu and P. Samarati. 1994. Access control: principle and practice. *IEEE Communications Magazine*, 32, 9, 40–48. doi:10.1109/35.312842.

[22] Ravinderpal Singh Sandhu. 1992. Undecidability of safety for the schematic protection model with cyclic creates. *Journal of Computer and System Sciences*, 44, 1, 141–159. doi:10.1016/0022-0000(92)90008-7.

[23] J.A. Solworth and R.H. Sloan. 2004. A layered design of discretionary access controls with decidable safety properties. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE, Berkeley, CA, USA, 56–67. ISBN: 978-0-7695-2136-7. doi:10.1109/SECPRI.2004.1301315.

[24] Masakazu Soshi. 2000. Safety Analysis of the Dynamic-Typed Access Matrix Model. In *Computer Security - ESORICS 2000*. Frédéric Cuppens, Yves Deswarte, Dieter Gollmann, and Michael Waidner, (Eds.) Springer, Berlin, Heidelberg, 106–121. ISBN: 978-3-540-45299-7. doi:10.1007/10722599_7.

[25] Masakazu Soshi, Mamoru Maekawa, and Eiji Okamoto. 2004. The dynamic-typed access matrix model and decidability of the safety problem. *IEICE Transactions on Fundamentals*, E87-A, 1, 190–203.

[26] Mahesh V. Tripunitara and Ninghui Li. 2013. The Foundational Work of Harrison-Ruzzo-Ullman Revisited. *IEEE Transactions on Dependable and Secure Computing*, 10, 1, 28–39. doi:10.1109/TDSC.2012.77.

[27] Vincent Vatter. 2011. Maximal Independent Sets and Separating Covers. *The American Mathematical Monthly*, 118, 5, 418. doi:10.4169/amer.math.monthly.118.05.418.

[28] David R. Wood. 2011. On the number of maximal independent sets in a graph. *Discrete Mathematics & Theoretical Computer Science*, Vol. 13 no. 3, Combinatorics, 543. doi:10.46298/dmtcs.543.