Proc. of 7th IEEE Computer Security Foundations Workshop, Franconia, NH, June 1994, pages 138-149.

One-Representative Safety Analysis in the Non-Monotonic Transform Model

Paul E. Ammann and Ravi S. Sandhu*

Center for Secure Information Systems &
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

Abstract

We analyze the safety question for the Non-Monotonic Transform (NMT) model, an access control model that encompasses a wide variety of practical access control mechanisms. In general, safety analysis, i.e *whether it is possible for a specified subject to obtain a given access right for a certain object, is computationally intractable, even for many monotonic models. We identify one-representable NMT schemes and argue that they have tractable safety analysis. Safety analysis of one-representable schemes considers exactly one representative of each type of subject in the initial state, and thus the complexity of safety analysis is independent of the total number of subjects in the system. We demonstrate by example that one-representable schemes admit applications of practical interest, and that safety analysis guides the construction of such schemes.*

1 Introduction

The concept of transformation of access rights was introduced in its monotonic form by Sandhu [San89]. Monotonic transformations only add access rights in the system, but do not remove previously existing rights. Sandhu and Suri [SS92] extended the concept to include non-monotonic transformations, i.e., transformations which not only add access rights in the system, but in the process may also remove existing rights.

Transformation of rights unifies a surprising variety of access-control mechanisms found in the literature. Monotonic transformations include mechanisms such as amplification, copy flags, synergistic authorization, and some common forms of separation of duties [San89]. Non-monotonic transformations can account for transfer-only and countdown privileges, among others [SS92]. If these various mechanisms were to be lumped together, the result would be a complex *ad hoc* model in totality. Instead it has been demonstrated that a few basic concepts, namely, strong typing, grant transformations and internal transformations, suffice to express all these mechanisms and more.

Sandhu and Suri [SS92] defined a formal model called NMT (for Non-Monotonic Transform) which incorporates these concepts. They outlined a simple implementation of NMT in a distributed environment, using the familiar client-server architecture.

This paper focuses on safety analysis in NMT. The safety problem in access control asks whether or not a given subject can ever acquire a particular access right to a given object. For monotonic transformations, safety can be decided in polynomial time [San89]. NMT is known to have decidable safety. However, the reduction given in [SS92] is to a problem which is exponentially hard in the number of subjects, and is to a model [LS78] that is much more general than NMT.

The principal contribution of this paper is the identification of a class of NMT schemes called one-representable, for which safety is computationally tractable. Intuitively speaking, these schemes allow safety analysis to be accomplished by introducing exactly one subject of each type. This subject serves as a representative for all instances of that type. In this manner the complexity of safety analysis is independent of the actual number of subjects in the system. All monotonic schemes are one-representable. Our contribution is to identify some simple sufficient conditions which guarantee one-representability for non-

*The work of both authors is partially supported by National Science Foundation grant CCR-9202270. Ravi Sandhu is also supported by the National Security Agency through contract MDA904-92-C-5141. We are grateful to Dorothy Darnauer, Nathaniel Macon, Howard Stainer, and Mike Ware for their support and encouragement in making this work possible.

monotonic schemes. We demonstrate, by examples, that one-representable schemes admit applications of practical interest, and that safety analysis is useful in the design of such schemes.

The safety analysis problem has been a long-standing barrier to progress in the area of access control systems, which are flexible and can be customized to enforce a specific organization's policies. The negative results of [HRU76, HR78] showed that safety is undecidable under surprisingly weak assumptions. Since then progress [AS92, San88, Sny81] has been made on the safety problem for cases which are monotonic, or can be treated as such for safety analysis purposes. Results for non-monotonic cases have been negative [Bud83, LS78], underscoring the computational difficulty of this problem. The positive results of this paper stand in contrast to the past, and offer a new hope for progress in this important area.

The rest of this paper is organized as follows. Section 2 outlines the non-monotonic transform model. Section 3 defines the concept of one-representable schemes in NMT, and provides sufficient conditions for schemes that satisfy this property. Section 4 discusses the complexity of safety analysis for one-representable NMT schemes. Section 5 illustrates, by examples, how our safety analysis results are useful in the design of NMT systems. Section 6 concludes the paper.

2 The NMT model

In this section we briefly define the *Non-Monotonic Transform* (NMT) model. Detailed motivation of and discussion about the model, as well as an implementation outline, can be found in [SS92].

The protection state in NMT can be viewed in terms of the familiar access matrix. There is a row for each subject in the system, and a column for each object. In NMT the subjects and objects are disjoint. NMT does not define any access rights for operations on subjects, which are assumed to be completely autonomous entities.

NMT consists of a small number of basic constructs, and a language for specifying the commands which cause changes in the protection state. For each command we have to specify the authorization required to execute that command, as well as the effect of the command on the protection state. NMT provides three kinds of commands for changing the protection state, as follows:

1. Creation commands, which allow subjects to create objects.
2. Internal transformation commands, which allow a subject that possesses certain rights for an object to obtain additional rights. In the course of doing so the subject may lose one or more rights previously possessed by the subject.
3. Grant transformation commands, which enable granting of access rights by one subject to another. The general idea is that possession of a right for an object by a subject allows that subject to give some other rights for that object to another subject. Again, in the course of this process the first subject may lose one or more rights previously possessed by that subject.

The language for specifying these commands is defined below. (The syntax for NMT commands presented here is more compact than that in [SS92].)

2.1 Rights

Each system has a set of rights R. It is important to understand that R is not specified in the model but varies from system to system. We will generally expect R to include the usual rights such as *own*, *read*, *write*, *append*, and *execute*. But this is not required by the model. We also expect R to generally include more complex rights, such as *review*, *release*, etc. The meaning of these rights will be explained wherever they are used in our examples.

The access rights serve two purposes. Firstly, presence of a right, such as *read*, in the $[S, O]$ cell of the access matrix authorizes S to perform the read operation on O. Secondly, presence of a right, say *own*, in $[S, O]$ authorizes S to perform some operation which changes the access matrix, e.g., by entering *read* in $[S', O]$. In other words, S as the owner of O can change the permissions in the access matrix so that S' can read O. The focus of NMT is on this second purpose of rights, i.e., the authorization by which the access matrix itself gets changed.

2.2 Types of subjects and objects

The notion of *protection type*, or simply *type*, is fundamental to NMT. All subjects and objects are assumed to be strongly typed. Strong typing requires that each subject or object is created to be of a particular type, which thereafter cannot change.

NMT requires that a disjoint set of subject types, TS, and object types, TO, be specified in a scheme. For example, we might have TS={*user, security-officer*} and TO={*user-files, system-files*}, with the significance of these types indicated by their names.

2.3 Creation commands

Object creation is specified by a finite number of creation commands, each of which has the following format.

```
$\textbf{CREATE}_i(S{:}u,\ O{:}o) \equiv$
        $\textbf{create}\ O;\ [S,O] := Y;$
```

This command is interpreted as saying that a subject S of type u can create an object O of type o. The effect of creation is that the creator gets $Y = \{y_1, \ldots, y_m\}$ rights for the newly created object. In terms of the access matrix, a new column for O is created with all cells empty except for $[S, O]$, which contains the rights $\{y_1, \ldots, y_m\}$.

The i in the command name is an identifier used to distinguish among different **CREATE** commands. Typically i will be a small integer or a symbolic name. The **GRANT** and **ITRANS** commands, defined below, are similarly distinguished by attaching an identifier to make each name unique.

2.4 Grant transformation commands

A grant transformation command has the general format given below.

```
$\textbf{GRANT}_i(S_1{:}u,\ S_2{:}v,\ O{:}o) \equiv$
    $\textbf{if}\ X \subseteq [S_1,O]\ \textbf{then}$
        $\begin{cases} [S_1,O] := [S_1,O] - Y; \\ [S_2,O] := [S_2,O] \cup Z; \end{cases}$
```

Here X, Y, and Z are subsets of R. S_1 is the *source* of the operation and S_2 is the *destination*. A grant transformation is interpreted as follows: subject S_1 of type u can grant $Z = \{z_1, \ldots, z_l\}$ rights for an object O of type o to subject S_2 of type v provided S_1 has $X = \{x_1, \ldots, x_n\}$ rights for O, but in the transformation S_1 will lose the $Y = \{y_1, \ldots, y_m\}$ rights for O.* A monotonic grant transformation is a special case of our **GRANT** command, in which Y is empty.

2.5 Internal transformation commands

An internal transformation command has the general format given below.

```
$\textbf{ITRANS}_i(S{:}s,\ O{:}o) \equiv$
    $\textbf{if}\ X \subseteq [S,O]\ \textbf{then}$
        $[S,O] := [S,O] - Y \cup Z;$
```

*The definition of NMT in [SS92] also requires $Y \subseteq X$; in essence an operation cannot delete a right from the source unless the right is actually present in the source. Requiring $Y \subseteq X$ corresponds to the definition of normal operations introduced in section 3.

As in grant transformation, X, Y, and Z are subsets of R. The interpretation of this command is that a subject S of type u who has the $X = \{x_1, \ldots, x_n\}$ rights for an object O of type o can obtain the $Z = \{z_1, \ldots, z_l\}$ rights for O by internal transformation, but in the process S will lose the rights $Y = \{y_1, \ldots, y_m\}$ for O.† If the Y set of rights is empty, the internal transformation reduces to its monotonic case.

2.6 Summary

A summary of the NMT model is as follows:

Definition 1 A *scheme* for transformation of rights is stated in NMT by specifying the following finite components.

1. A set of rights R.
2. Disjoint sets of subject types TS, and object types TO.
3. A set of object creation commands $\{\textbf{CREATE}_i\colon i = 1 \ldots l\}$.
4. A set of grant transformation commands $\{\textbf{GRANT}_i\colon i = 1 \ldots m\}$.
5. A set of internal transformation commands $\{\textbf{ITRANS}_i\colon i = 1 \ldots n\}$.

Definition 2 An NMT *system* consists of an NMT scheme, an initial set of subjects and objects, and the initial access matrix.

Note that, although revocation commands are defined in NMT [SS92], we ignore them here since they do not affect worst-case safety analysis. Commands for destroying objects are similarly ignored.

It is clear that NMT treats each column of the access matrix independently of any other column. Thus, in analyzing the behavior of NMT it suffices to focus on one column at a time. This is in contrast to more general models, such as [AS92, HRU76, San88, San92], in which the state of one column can affect the behavior of commands on another column. NMT has been kept deliberately simple in this way, and yet it accommodates many practically useful access control policies as shown in [San89, SS92].

†The comment in the previous footnote applies here also.

3 One-representative schemes

In this section we address analyzability of NMT. In general, safety in NMT is decidable, but of exponential complexity in the number of subjects in the system. Since subject creation is not in the NMT model, the number of subjects technically is fixed for any given system and, as shown in [LS78], the restriction is sufficient to yield decidable safety.

The practical objection to this decidability result is that exponential complexity in the number of subjects is computationally too expensive. NMT is so much simpler than the general case of [LS78] that it seems intuitively reasonable to expect a more efficient safety analysis.

To exploit the simplicity of NMT, we look for restrictions that limit the number of subjects necessary for safety analysis. In this paper, we define restrictions that limit the number of necessary subjects to one of each protection type. Such a subject is referred to as a *representative*, because the subject can simultaneously represent any number of other subjects of the same type. Thus the total number of subjects has no effect on complexity, since only one subject of each type is required for analysis. These concepts are formalized in the following definitions.

Definition 3 A *representative* (with respect to some object O of type o) is a subject that is capable of mimicking the operations on O of one or more other subjects of the same type.

Definition 4 A representative is *blank* (with respect to an object O) if there are no access rights (for O) associated with that representative.

Definition 5 A scheme is *1-representative analyzable* (or *one-representative analyzable*) if the safety question can be answered by analyzing at most one representative of each protection type.

Recall that object creation in NMT gives rights for the object to exactly one subject and thus ties the creator of the object to the object itself. In general, there will be exactly one non-blank representative in the state immediately following any **CREATE** operation. Thus, if we let $n = |TS|$, the number of blank representatives needed for 1-representative analysis is at most $n - 1$.

The structure of the rest of this section is as follows. First, we use examples to illustrate two properties which are undesirable, from a safety perspective, in that the properties can lead to one representative being insufficient for safety analysis. We then formalize rules to exclude these properties and prove that these rules are sufficient to guarantee that a scheme is 1-representative analyzable.

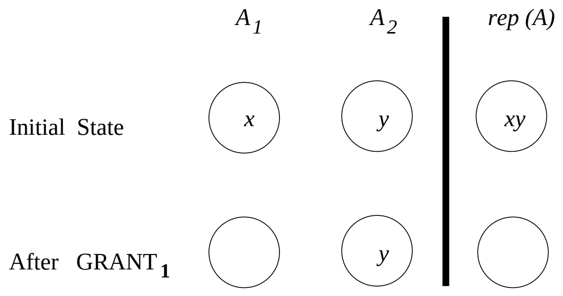

Figure 1: Effect of Non Normal Operation

We start with a distinction between rights that can affect the evolution of a protection state and those that cannot:

Definition 6 A *propagation right* x is a right where a test is made for the presence of x in the precondition of some **GRANT** or **ITRANS** operation.

In answering safety questions, propagation rights impact the analysis in a way that non-propagation rights cannot. Specifically, the presence or absence of propagation rights determines whether a specific **GRANT** or **ITRANS** operation can be carried out in a particular state.

3.1 Non-normal schemes

Consider the example illustrated in figure 1. The example shows one way that, for certain schemes, two subjects of a given type can accomplish more than a single representative that is trying to simulate the actions of both. In the example the two distinct subjects are A_1 and A_2 of type a. We suppose that initially $x \in [A_1, O]$, and $y \in [A_2, O]$. This is indicated in figure 1 by placing x and y respectively inside circles labeled A_1 and A_2. The identity of the object O is not explicitly shown in the figure. Consider the operation **GRANT**$_1$.

```
GRANT_1(A:a, B:b, O:o) ≡
    if $\{x\} \subseteq [A,O]$ then
        $\left\{ \begin{array}{l} [A,O] := [A,O] - \{x,y\}; \\ [B,O] := [B,O] \cup \{z\}; \end{array} \right.$
```

If **GRANT**$_1$ is applied to A_1 the effect of subtracting $\{x, y\}$ is to leave A_1 with the empty set of rights. A_2

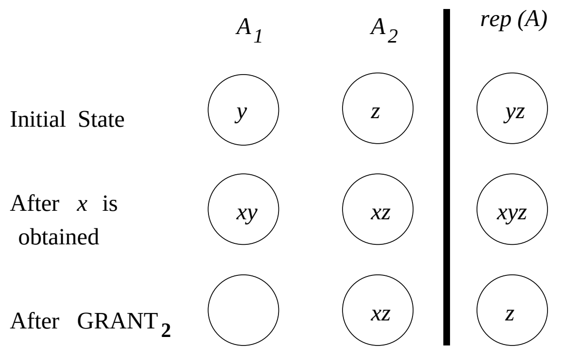

Figure 2: Effect of Duplicate Rights

is unaffected by the application of $\mathbf{GRANT}_1$ to A_1 and is therefore free to use y in some other **GRANT** or **ITRANS** operation.

Now consider what happens if we use a single subject, *rep*(A) to try to simulate A_1 and A_2. The initial state of *rep*(A) is the union of the states of A_1 and A_2, or $\{x, y\}$. After operation $\mathbf{GRANT}_1$ is applied to *rep*(A), the state of *rep*(A) is the empty set. But then *rep*(A) is no longer able to simulate the actions of A_2, since A_2 holds y and *rep*(A) does not.

The problem in this example is that $\mathbf{GRANT}_1$ deleted a (propagation) right y from the source even though there was no test for y in the precondition. We formalize this notion as follows:

Definition 7 An operation is *normal* if the precondition of the operation tests for the presence of every propagation right that is deleted from the source cell. Otherwise the operation is *non-normal.* A scheme is *normal* if all operations in the scheme are normal.

3.2 Duplicate schemes

Consider a second example, illustrated in figure 2. The example shows another way that, for certain schemes, two subjects of a given type can accomplish more than a single representative that is trying to simulate the actions of both. In the example the two distinct subjects are A_1 and A_2 of type a. We suppose that the initial state of A_1 is y, and the initial state of A_2 is z. Suppose that A_1 and A_2 can both obtain the right x via some unspecified transform operations in the scheme. Now consider $\mathbf{GRANT}_2$, which is a normal operation.

```
GRANT_2(A:a, B:b, O:o) ≡
    if $\{x, y\} \subseteq [A,O]$ then
        $\begin{cases} [A,O] := [A,O] - \{x, y\}; \\ [B,O] := [B,O] \cup \{p\}; \end{cases}$
```

Suppose $\mathbf{GRANT}_2$, is applied with A_1 as the source. Although A_1 is subsequently empty, A_2 still holds x and z, and some other unspecified transform operation that tests for $\{x, z\}$ in the precondition is applicable to A_2.

Now consider what happens if we use a single subject, *rep*(A) to try to simulate the effects of A_1 and A_2. The initial state of *rep*(A) is the union of the states of A_1 and A_2, or $\{y, z\}$. Subsequently, *rep*(A) acquires x, yielding a state $\{x, y, z\}$. Note that *rep*(A) may acquire x twice, but only a single copy of a right can be maintained since adding a right to a cell is defined via set union.

The representative *rep*(A) can carry out $\mathbf{GRANT}_2$, but is then not able to represent the behavior of A_2, since the right x is consumed by $\mathbf{GRANT}_2$. The problem is that a subject could obtain a right, x in this case, even though that subject already holds x. Specifically, it is possible the *rep*(A) first applied the operation that A_1 used to obtain x, and the applied the operation that A_2 used to obtain x. In the second of these steps, the duplicate right x is absorbed by *rep*(A), and the subsequent evolution of the protection state is curtailed. We formalize this property below.

Definition 8 A right x is *monotonic* if x is a propagation right, and if x may not be removed from a cell once x has been introduced. A right x is *non-monotonic* if x is a propagation right, and x may be removed from a cell by a **GRANT** or **ITRANS** operation.

Definition 9 A scheme is *non-duplicate* if it is never possible to introduce a non-monotonic right x into a cell if x is already present in that cell. Otherwise a scheme is said to be *duplicate.*

3.3 Normal, nonduplicate schemes

The main result of this paper is that certain schemes, namely those that are normal and non-duplicate, have a simpler, more tractable safety analysis than more general schemes. Specifically, schemes that conform to these properties are 1-representative analyzable. The intuitive reason for this result is that the situation depicted in figures 1 and 2 cannot arise in normal, non-duplicate schemes. We need to show that avoiding these situations is sufficient to achieve 1-representative analyzability.

We begin by formalizing the notion of a history.

Definition 10 A *protection state* (with respect to some object O) is a tuple with an entry for each subject. Each tuple entry is the set of rights (for O) held by the corresponding subject.

Definition 11 A *history* for a scheme is a finite sequence of protection states. The first state in any history is that produced by the initial object creation operation. Each subsequent state is related to the immediately preceding state by some **GRANT** or **ITRANS** operation.

We construct a 1-representative history H' from H by replacing the source and destination subjects in each operation in H with the representative subject of the appropriate type. As demonstrated in figures 1 and 2, in general, some of the **GRANT** or **ITRANS** operations in H' may fail. In such a case, we say that history H' is invalid; otherwise H' is valid. Our goal is to show that for normal, non-duplicate schemes, all possible histories H' are valid. Next, we develop a predicate on histories that ensures validity.

The *partition predicate* is that, for all histories, H, where $|H| = n$, all states in H partition the states in H' (with respect to non-monotonic rights), where H' is constructed from H as described above.

Let $H(k) \cdot s$ be the set of non-monotonic rights held by subject s of type t in state k of history H. Similarly, let $H'(k) \cdot rep(t)$ be the set of non-monotonic rights held by the type t representative in state k of history H'. Formally, the partition predicate in state k, where k is an index into the histories H and H', is $P(k)$ as defined below.

$$(\forall t \in TS)[H'(k) \cdot rep(t) = \bigcup_{type(s)=t} H(k) \cdot s] \ \wedge$$

$$(\forall s_1,\ s_2 \ \mid\ type(s_1) = type(s_2) \wedge s_1 \neq s_2)$$
$$[H(k) \cdot s_1 \cap H(k) \cdot s_2 = \emptyset]$$

(Recall that TS is the set of subject types.)

The partition predicate yields the property that for any operation *op* whose precondition is satisfied for some subject S after k operations in history H, it is the case that the precondition of *op* is satisfied for the representative of S, $rep(S)$, after k operations in history H'.

If the partition predicate holds for all states in a history H, then H' is also a valid history and a single representative is sufficient for analysis. We may answer a safety question for an arbitrary cell of type t by examining the representative of type t in H'. Further, we may answer a safety question about a combination of cells, each of a distinct type, by examining the corresponding representatives in H'.

In the following theorem, we prove that normal, non-duplicate scheme are 1-representative analyzable, because they preserve the partition predicate.

Theorem 1 A normal, non-duplicate scheme is 1-representative analyzable.

Proof: From the above discussion, it follows that it suffices to show that the partition predicate holds for all states in an arbitrary history H. We proceed by induction over the length of such an H for a normal, non-duplicate scheme. The predicate for induction is the partition predicate.

Basis Case ($n = 1$): The first operation in H must be a **CREATE** operation. The state produced by object creation satisfies the partition predicate, since all cells are empty except for the cell associated with the subject that carried out the creation. Since H' and H are identical, the partition predicate is satisfied for all histories of length 1.

Inductive Step: We assume that for all histories H_k, where $|H_k| = k$, the partition predicate is satisfied. We show that for all histories H_{k+1}, where $|H_{k+1}| = k + 1$, the partition predicate is still satisfied. Since each history of length $k + 1$ is obtained by adding one operation to some history of length k, the inductive hypothesis assures us for the first k operations in H_{k+1}, the partition predicate is satisfied.

The last state in H_{k+1}, denoted $H_{k+1}(k + 1)$, is obtained by applying some operation *op* to the next to last state in H_{k+1}, which is $H_{k+1}(k)$. By the induction hypothesis, the partition predicate applies to $H_{k+1}(k)$, so we may replace the source and destination cells with the appropriate representatives and apply *op* to $H'_{k+1}(k)$. We are obliged to show that the partition predicate is satisfied with respect to $H_{k+1}(k + 1)$ and $H'_{k+1}(k + 1)$.

Suppose that *op* is a **GRANT** operation. Let A be the source and B be the destination of *op* in H (note that A may equal B). Let R_s be the set of rights deleted from the source A and R_d be the set of rights added to the destination B. Since the scheme is normal, $R_s \subseteq H_{k+1}(k) \cdot A$. Thus subtracting R_s from $H(k) \cdot A$, and from $H'(k) \cdot rep(A)$, maintains the partition predicate in state $k + 1$. Since the scheme is guaranteed to be non-duplicate, no non-monotonic right x, $x \in R_d$ is already present in any cell C, where the types of B and C match. To see this, let C, instead of B, be the destination of *op*. The operation *op* still applies, since the precondition tests A, but not B. In this case, C can acquire x even though it already holds x, and the non-duplicate property is violated. Thus we may add R_d to both B in $H(k)$ and $rep(B)$ in $H'(k)$ and maintain the partition predicate.

Suppose *op* is an **ITRANS** operation, and let A be the source (and destination) of the operation in H. Define R_s and R_d as above. The same analysis for R_s applies for **ITRANS** operations as for **GRANT** operations. The analysis for R_d differs. Let $B \neq A$ be another subject in H of the same type as A. On first inspection, it might appear that because the precondition on *op* can be satisfied in A but not in B, A may obtain x via *op* while B already holds x. Thus in the final state, the partition predicate would not be satisfied since A and B both hold x. However, such a scenario is impossible, as we show next.

Since the partition predicate holds on any history of k states (by the induction hypothesis), the first k elements of H'_{k+1} form a valid history, which we denote H'_k. Thus we may apply *op* to $rep(A)$ in the last state of H'_k. Suppose that $rep(B)$ already holds x, as assumed above. If *op* introduces x into $rep(A)$, then, since $rep(A) = rep(B)$ (recall, A and B are of the same type), the representative is obtaining x even though it already holds x, and the non-duplicate property is violated, which is a contradiction. Thus the partition predicate is maintained by **ITRANS** operations. □

4 One-representative safety analysis

This section gives the complexity of safety analysis under the normal, non-duplicate restrictions and argues that safety is tractable for cases of practical interest.

If a scheme is one-representative analyzable, a direct algorithm to answer the safety question is to start with the initial state for the subject that created the object, augment the initial state with a representative of each unrepresented protection type, and compute the states that are reachable from the augmented state. If there are $n = |T|$ protection types and $r = |R|$ rights, there are at most 2^{rn} possible protection states. The worst case execution time is thus $\mathbf{O}(2^{rn})$.

The effect of the analysis bound is improved by several observations. First, it is not all rights, R, but only the non-monotonic rights that lead to a state explosion. Analysis for monotonic rights requires polynomial instead of exponential complexity in the number of rights. Also, non-propagation rights do not affect the reachability of a given protection state.

Next, it may be that the average time is substantially better than the worst case time for schemes of practical interest. Although the examples in the next section support this conjecture, it is likely that empirical study is required to determine typical execution times.

But the key observation is that if we evaluate complexity in the total number of possible subjects and treat the scheme as a constant, 1-representative analyzability is $\mathbf{O}(1)$. The coefficient for the analysis is (2^{rn}), but both the number of rights and the number of types are fixed (constant) for a given scheme. The great advantage of 1-representative analyzability is that we need not worry about multiple subjects of the same type.

Although we have eliminated the number of possible subjects as a factor, it is useful to see whether a more efficient coefficient than (2^{rn}) is feasible. Our next results bounds any possible improvement to (2^n) (barring any progress on reducing NP to P).

It turns out that the safety question for schemes which are non-duplicate and normal, (and thus for schemes which are one-representative analyzable,) is NP-hard in n, the number of types. Thus it is not reasonable to expect to find algorithms for such schemes with significantly better worst case times. (Of course, there may be further restrictions on schemes which do result in a simpler analysis). We show this result by reducing 3-satisfiability (3-sat) to the safety problem.

Theorem 2 3-sat reduces to the safety problem in normal, non-duplicate NMT schemes.

Proof Sketch: We do not give a complete proof, but only a general idea. To encode 3-sat in a non-duplicate, normal transform scheme, we proceed as follows. There are m variables in a 3-sat problem; and a literal, *i.e.* a variable or its negation, may appear in some subset of n terms. Each term consists of a disjunction (i.e., logical OR) of exactly 3 literals. The satisfiability predicate is the conjunction (i.e., logical AND) of all n terms.

We represent each variable by one instance of a distinct type. We represent each term by one instance of a distinct type. We represent the satisfiability predicate itself with one instance of a distinct type. Thus the scheme employs $m + n + 1$ types. We encode "true" and "false" as two distinct rights. We encode a "before" and "after" program counter as two distinct rights. The initial state is that there is a "before" program counter in the subject corresponding to the first variable. Each subject corresponding to a variable makes a nondeterministic choice (via selecting amongst two **ITRANS** operations) that consumes the "before" counter and produces either a "true" or "false" for the variable and an "after" counter. The "after" counter is communicated (via a **GRANT**) to

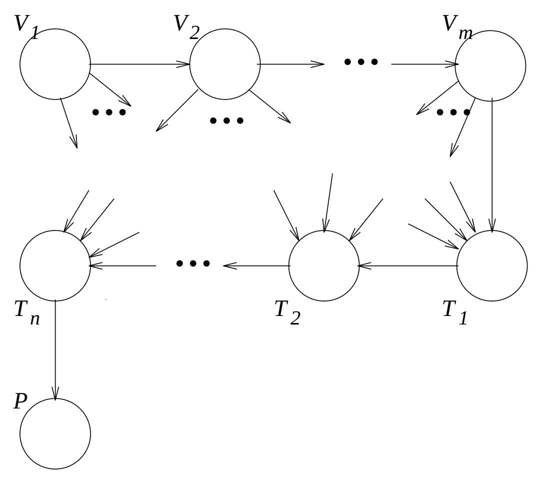

Figure 3: 3-Sat as Safety: Can P Acquire The Right "true"?

the subject representing the next variable. **GRANT** operations are defined that allow the selection between "true" and "false" to be communicated to the appropriate subjects representing terms. The last variable subject grants a "before" token to the first term subject. If rights encoding sufficient literals are present in a term subject, the term subject grants a "before" program counter to the next term subject, or, in the case of the last term subject, a "true" to the predicate subject.

The construction is broadly illustrated in figure 3, where the V_i are cells implementing variables, the T_j are cells implementing terms, and P is a cell implementing the satisfiability predicate. Arcs represent the transfer of rights via various operations; the linear chain of arcs represents the flow of "program counter" rights, and the the arcs that fan out from the V_i represent the flow of literals to the appropriate terms. Each term T_j has exactly 3 incoming arcs, each of which represents one of the literals in that term.

The safety question is, can the right "true" appear in the node representing the predicate. Safety analysis yields "yes" iff the corresponding predicate is satisfiable. Thus 3-satisfiability reduces to safety. $\Box$

One final, unfortunate observation may be made on the scheme used in the construction: the scheme is acyclic, in that there is no sequence of operations such that the destination of each operation matches the source of the next, and the destination of the last operation matches the source of the first. Thus restricting transform schemes to be acyclic is not sufficient to keep the analysis from being NP-hard.

5 Document release examples

In this section we give variations on a document approve/release example to show how safety analysis can be applied to guide the implementation of various polices in NMT. We also see via example how far from the worst case the actual complexity of safety analysis can be for a practical system.

We consider the case of a scientist who creates a document and consequently gets the *own*, *read*, and *write* privileges for it. We stipulate that prior to releasing this document for publication, the scientist needs approvals from two separate and independent sources. The security-officers and the patent-officers of the organization are the two types of users who can grant the scientist each of these separate approvals. They, of course, need to review the document before granting approval.

After preparing the document for publication, the scientist asks for review from a security-officer and a patent-officer. In the process, the scientist loses the *write* privilege to the document, since it is clearly undesirable for a document to be edited during or after a (successful) review.

After review of the document, the security-officer and the patent-officer each grant the scientist an appropriate approval. It is reasonable to disallow further attempts to review the document after an approval is granted. Thus the *review* privilege for the document is lost as approval is granted. After obtaining approval from both officers, the scientist can internally transform the approvals into the *release* privilege needed to publish the document.

To express this policy, and the variations that follow, we employ the following rights and types:

1. $R = \{$*own*, *read*, *write*, *ask-sec*, *ask-pat*, *review*, *sec-ok*, *sec-reject*, *pat-ok*, *pat-reject*, *release*$\}$
2. $TS = \{sci,\ so,\ po\}$, $TO = \{doc\}$

Briefly, many rights correspond to stages in the approval process: *ask-pat* is the right to ask the patent officer for a review, *review* is the right that lets an officer review a document, *pat-ok* is the right that is returned if the patent review is satisfactory, and so on. Some examples do not use all of the rights, but it is convenient to define the rights and types once and then use an appropriate subset. Subjects types are abbreviations for scientist, security-officer, and patent-officer.

The same creation operation is used in all the examples. We present it once:

CREATE$_{doc}(S{:}sci,\ O{:}doc) \equiv$
create O; $[S,O] := \{own,\ read,\ write\}$;

An initial specification of the problem, obtained from the English description above, might be as follows.

Scheme 1 Initial Specification

```
$\mathbf{GRANT}_{seek\cdot security\cdot ok}(S_1{:}sci,\ S_2{:}so,\ O{:}doc) \equiv$
    if $own \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - write; \\ [S_2,O] := [S_2,O] \cup review;\end{cases}$

$\mathbf{GRANT}_{seek\cdot patent\cdot ok}(S_1{:}sci,\ S_2{:}po,\ O{:}doc) \equiv$
    if $own \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - write; \\ [S_2,O] := [S_2,O] \cup review;\end{cases}$

$\mathbf{GRANT}_{approve\cdot sec}(S_1{:}so,\ S_2{:}sci,\ O{:}doc) \equiv$
    if $review \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup sec\text{-}ok;\end{cases}$

$\mathbf{GRANT}_{approve\cdot pat}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    if $review \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup pat\text{-}ok;\end{cases}$

$\mathbf{ITRANS}_{get\cdot release}(S{:}sci,\ O{:}doc) \equiv$
    if $own,\ sec\text{-}ok,\ pat\text{-}ok \subseteq [S,O]$ then
        $[S,O] := [S,O] - sec\text{-}ok,\ pat\text{-}ok \cup release;$
```

We describe the scheme as follows. The scientist can create the document and edit it via the *read* and *write* privileges. By using the $\mathbf{GRANT}_{seek\cdot security\cdot ok}$ and $\mathbf{GRANT}_{seek\cdot patent\cdot ok}$ operations, the scientist can request approval from a security-officer and a patent-officer, respectively. In doing so, the scientist loses the *write* privilege. After review, these officials employ the $\mathbf{GRANT}_{approve\cdot sec}$ and $\mathbf{GRANT}_{approve\cdot pat}$ operations to give the scientist the necessary rights to use $\mathbf{ITRANS}_{get\cdot release}$ to obtain the *release* privilege.

An initial observation on the scheme is that the operations that grant the *review* right to the security and patent officers are potentially non-normal. However, in this scheme, the *write* privilege is not a propagation right, and so the consideration of normal versus non-normal does not apply. The specification has the desirable property that it does not matter if approval is sought first from the security-officer or first from the patent-officer. In either case, the *write* privilege is lost immediately.

It may appear from the English specification, and from an initial scan that the scheme is also non-duplicate. For example, the *review* right, which is clearly non-monotonic, is consumed by the patent-officer while granting the scientist the *pat-ok* right, and thus the $\mathbf{GRANT}_{approve\cdot pat}$ operation consumes its precondition.

However, if the analysis procedure of the previous section is applied, it reveals that the scheme *is* duplicate. The scientist is free to ask for an arbitrary number of reviews, and, in particular, the scientist can grant the *review* right to a reviewer even if the reviewer already holds the *review* right. One option out of this dilemma is to make a policy decision that multiple *review* rights are acceptable. Under this interpretation, the given specification is misleading since rights that are apparently destroyed can easily be recovered. Since the scheme is duplicate, it is also unanalyzable via the techniques developed in this paper. An alternate specification of such a policy is discussed at the end of this section.

For our present purposes, we follow the English specification more closely, and adopt the policy that the scientist cannot have more than one security review request (or more than one patent review request) outstanding. To this end, we modify the scheme so that it is non-duplicate. We make the *write* privilege a propagation right and link consumption of the *write* privilege with the granting of rights to ask for review. We add one new operation, $\mathbf{ITRANS}_{finish\cdot document}$, and redefine two others, $\mathbf{GRANT}_{seek\cdot security\cdot ok'}$ and $\mathbf{GRANT}_{seek\cdot patent\cdot ok'}$. (In general, redefinition is denoted by a trailing tick on the operation name.) Other operations are unaffected and so are not explicitly relisted below, although the unaffected operations *are* part of the scheme.

Scheme 2 Non-Duplicating Specification (Modifies Scheme 1)

```
$\mathbf{ITRANS}_{finish\cdot document}(S{:}sci,\ O{:}doc) \equiv$
    if $own,\ write \subseteq [S,O]$ then
        $[S,O] := [S,O] - write \cup ask\text{-}sec,\ ask\text{-}pat;$

$\mathbf{GRANT}_{seek\cdot security\cdot ok'}(S_1{:}sci,\ S_2{:}so,\ O{:}doc) \equiv$
    if $ask\text{-}sec \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - ask\text{-}sec; \\ [S_2,O] := [S_2,O] \cup review;\end{cases}$

$\mathbf{GRANT}_{seek\cdot patent\cdot ok'}(S_1{:}sci,\ S_2{:}po,\ O{:}doc) \equiv$
    if $ask\text{-}pat \subseteq [S_1,O]$ then
        $\begin{cases}[S_1,O] := [S_1,O] - ask\text{-}pat; \\ [S_2,O] := [S_2,O] \cup review;\end{cases}$
```

The scheme is now one-representative analyzable. An enumeration by hand of all reachable states results in 11 states being generated. This is insignificant when compared to the theoretical worst case of

2^{nr}, where n, the number of types, is 3 and r, the number of propagation rights, is at least 7. At least two informal observations help explain the small number of actual states. First, there aren't many possible successor states for any given state. Each operation is designed to apply to a small number of states. Second, the operations in this scheme use rights in very limited ways, and hence it is clear from inspection that many states are unreachable. For example, it is clear from a static observation of the operations that a security-officer can never obtain the *ask-pat*, *ask-sec*, or *release* rights. The document release example is encouraging in suggesting that, for practical applications, one-representative analyzability may have expected computational complexity that is much smaller than the worst case bound.

The next variation on the example is to allow for the security officer or the patent-officer to explicitly reject the document instead of simply withholding approval. One reasonable policy decision would be for the document to then be considered "dead". The scientist would be forced to create a new document and start over. We specify this extension below by adding two operations that can explicitly send rejection rights, *sec-reject* and *pat-reject* to the scientist. (Again, unaffected operations are not explicitly listed but are part of the scheme).

Scheme 3 Explicitly Specifying Rejection (Modifies Scheme 2)

```
$\mathbf{GRANT}_{reject\cdot sec}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup sec\text{-}reject; \end{cases}$

$\mathbf{GRANT}_{reject\cdot pat}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup pat\text{-}reject; \end{cases}$
```

Safety analysis shows Scheme 3 to be normal and non-duplicate and hence 1-representative analyzable. The number of reachable states has increased modestly to 18.

Instead of beginning a new document following a security or patent rejection, it might be more efficient, from the viewpoint of the scientist, to be allowed to edit the existing document. Since editing requires the write privilege, we might try to modify the operations $\mathbf{GRANT}_{reject\cdot sec}$ and $\mathbf{GRANT}_{reject\cdot pat}$ to allow for further editing by the scientist. After editing, the scientist can resubmit the document for review using previously defined operations. To this end, we replace the explicit rejection rights, *sec-reject* and *pat-reject*, with the rights to ask for review, *ask-sec* and *ask-pat* and obtain the following scheme. Note that at some modest increase in complexity, it would be possible to continue to use explicit rejection rights; for simplicity we choose not to do so here.

Scheme 4 First Attempt To Allow Re-Editing (Modifies Scheme 3)

```
$\mathbf{GRANT}_{reject\cdot sec'}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup ask\text{-}sec,\ write; \end{cases}$

$\mathbf{GRANT}_{reject\cdot pat'}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup ask\text{-}pat,\ write; \end{cases}$
```

Safety analysis now reveals that the scheme is duplicate; the scientist can receive the *write* privilege while holding the *write* privilege. Worse, even with one representative of each type, it is possible for the scientist to hold the *write* privilege for the document after the security-officer, the patent-officer, or both complete a review. It is also possible for the scientist to hold the *write* and *release* privileges simultaneously. The problem is that the exclusion between holding the *write* privilege and the *ask-sec* and *ask-pat* privileges has been broken.

We repair the scheme by further revising the definitions $\mathbf{GRANT}_{reject\cdot sec''}$ and $\mathbf{GRANT}_{reject\cdot pat''}$. (Note that the doubly revised operations are denoted with two trailing ticks). We add a new operation that is the inverse of the finish-document operation. To allow for undoing approval from one review authority to accommodate required revisions from the other review authority, we introduce two internal transformations that convert approval back into the right to ask for approval. The modified and new operations are:

Scheme 5 Analyzable Scheme That Allows Re-Editing (Modifies Scheme 3)

```
$\mathbf{GRANT}_{reject\cdot sec''}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup ask\text{-}sec; \end{cases}$

$\mathbf{GRANT}_{reject\cdot pat''}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - review; \\ [S_2,O] := [S_2,O] \cup ask\text{-}pat; \end{cases}$
```

```
$\mathbf{ITRANS}_{revise \cdot document}(S{:}sci,\ O{:}doc) \equiv$
    **if** $own,\ ask\text{-}sec,\ ask\text{-}pat \subseteq [S,O]$ **then**
        $[S,O] := [S,O] - ask\text{-}sec,\ ask\text{-}pat \cup write;$

$\mathbf{ITRANS}_{undo \cdot security \cdot ok}(S{:}sci,\ O{:}doc) \equiv$
    **if** $own,\ sec\text{-}ok \subseteq [S,O]$ **then**
        $[S,O] := [S,O] - sec\text{-}ok \cup sec\text{-}ask;$

$\mathbf{ITRANS}_{undo \cdot patent \cdot ok}(S{:}sci,\ O{:}doc) \equiv$
    **if** $own,\ pat\text{-}ok \subseteq [S,O]$ **then**
        $[S,O] := [S,O] - pat\text{-}ok \cup pat\text{-}ask;$
```

Analysis on this scheme reveals that there are 11 reachable states. In none of them is it possible for the scientist to hold the *write* privilege and a *sec-ok*, *pat-ok* or *release* privilege simultaneously.

Finally, we return to the question of analyzing a policy in which it is allowable for the scientist to ask for a review even though there is an outstanding request for a review. In this case, it is clearer to make the *review* right explicitly monotonic since, from a safety perspective, it is unnecessary to consider the deletion of a right if that right can be unconditionally restored. If a decision is made to make *review* monotonic, similar arguments apply, in this example, to make *sec-ok*, *pat-ok*, and *release* also monotonic. The full set of transformations for the scheme is:

Scheme 6 Revision Allowing Multiple Simultaneous Reviews (Replaces Scheme 1)

```
$\mathbf{GRANT}_{seek \cdot security \cdot ok}(S_1{:}sci,\ S_2{:}so,\ O{:}doc) \equiv$
    **if** $own \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - write; \\ [S_2,O] := [S_2,O] \cup review; \end{cases}$

$\mathbf{GRANT}_{seek \cdot patent \cdot ok}(S_1{:}sci,\ S_2{:}po,\ O{:}doc) \equiv$
    **if** $own \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - write; \\ [S_2,O] := [S_2,O] \cup review; \end{cases}$

$\mathbf{GRANT}_{approve \cdot sec}(S_1{:}so,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - ; \\ [S_2,O] := [S_2,O] \cup sec\text{-}ok; \end{cases}$

$\mathbf{GRANT}_{approve \cdot pat}(S_1{:}po,\ S_2{:}sci,\ O{:}doc) \equiv$
    **if** $review \subseteq [S_1,O]$ **then**
        $\begin{cases} [S_1,O] := [S_1,O] - ; \\ [S_2,O] := [S_2,O] \cup pat\text{-}ok; \end{cases}$

$\mathbf{ITRANS}_{get \cdot release}(S{:}sci,\ O{:}doc) \equiv$
    **if** $sec\text{-}ok,\ pat\text{-}ok \subseteq [S,O]$ **then**
        $[S,O] := [S,O] - \ \cup release;$
```

The safety analysis from the previous section can now be applied to the scheme since no non-monotonic right is duplicate. Analysis reveals a total of 10 reachable states. Inspection of reachable states verifies that in no state can the scientist simultaneously hold the *write* privilege and any of *sec-ok*, *pat-ok*, and *release*.

6 Conclusion

One approach to the safety problem for access control models is to identify constructs and restrictions that delimit special cases where the model is still practically useful but safety analysis is tractable. In this paper, we have identified restrictions that can make tractable the analysis for a subset of the non-monotonic transform model (NMT). Via examples, we have shown that practical polices can be expressed within these restrictions.

We consider schemes in which one (representative) subject of each type is sufficient to bound the worst-case evolution of the protection state, independent of the total number of subjects. We identify two properties, normality and non-duplication, and show that for schemes that satisfy both properties, one representative is sufficient for analysis. In addition, the failure of a scheme to adhere to either property can be detected during analysis.

Most importantly, the complexity of analysis is independent of the total number of subjects. The complexity of the analysis procedure given in this paper is exponential in the product of the number of types and the number of rights in the scheme under analysis. From a reduction of 3-satisfiability to safety, we have shown that the worst-case bound is unlikely to improve significantly. However, both the number of types and the number of rights are constant for any given scheme. Additionally, we conjecture that the analysis of many schemes of practical interest will in fact require substantially less effort than the worst-case behavior. This conjecture is supported by the document release examples given in section 5; further evidence likely requires empirical study.

The document release example of section 5 demonstrated how the analysis could reveal safety properties of practical schemes to guide implementation of various policies. The example demonstrated how the analysis procedure uncovers unexpected and/or undesirable properties of a given scheme and helps identify where the scheme should be altered.

The intractability of the safety analysis problem has been a long-standing barrier to progress in the area of access control systems, which are flexible and can be customized to enforce a specific organization's policies. The results in this paper are especially noteworthy in the absence of positive safety results for other non-monotonic models. These results are substantial progress towards overcoming the safety analysis barrier.

References

[AS92] P.E. Ammann and R.S. Sandhu. The extended schematic protection model. *The Journal Of Computer Security*, 1(3&4):335–384, 1992.

[Bud83] T.A. Budd. Safety in grammatical protection systems. *International Journal of Computer and Information Sciences*, 12(6):413–431, 1983.

[HR78] M.H. Harrison and W.L. Ruzzo. Monotonic protection systems. In R.A. DeMillo, D.P. Dobkin, A.K. Jones, and R.J. Lipton, editors, *Foundations of Secure Computations*, pages 337–365. Academic Press, 1978.

[HRU76] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[LS78] R.J. Lipton and L. Snyder. On snychronization and security. In R.A. DeMillo, D.P. Dobkin, A.K. Jones, and R.J. Lipton, editors, *Foundations of Secure Computations*, pages 367–385. Academic Press, 1978.

[San88] R.S. Sandhu. The schematic protection model: Its definition and analysis for acyclic attenuating schemes. *Journal of the ACM*, 35(2):404–432, April 1988.

[San89] R.S. Sandhu. Transformation of access rights. In *Proceedings IEEE Computer Society Symposium on Security and Privacy*, pages 259–268, Oakland, CA, May 1989.

[San92] R.S. Sandhu. The typed access matrix model. In *Proceedings IEEE Computer Society Symposium on Research In Security and Privacy*, pages 122–136, Oakland, CA, May 1992.

[Sny81] L. Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, C-30(3):172–181, 1981.

[SS92] R.S. Sandhu and G. Suri. Non-monotonic transformations of access rights. In *Proceedings IEEE Computer Society Symposium on Research In Security and Privacy*, pages 148–161, Oakland, CA, May 1992.