Logical method for reasoning about Access Control and Data Flow Control models

Luigi Logrippo

Département d'informatique et d'ingénierie
Université du Québec en Outaouais
Gatineau, QC, Canada
luigi@uqo.ca

Abstract. Some logic definitions for access control and data flow control models are proposed. A formalization of concepts of confidentiality and integrity is provided, on the basis of predicates CanKnow and CanStore. The application of these concepts to several well-known access control models, including Multi-Level Systems, Role-Based Access Control, Chinese Wall is shown. Formal definitions and proofs of invariant properties of these models in terms of our method will be given. It will then appear that these models have many possible variations and combinations of which only few have been studied. These concepts can be useful for developing proofs on access control models, automatically or manually, for developing new models, and for teaching access control and data flow control concepts.

Keywords: Access control models; flow control; formal methods, logic.

1 Introduction

In order to perform operations on data *objects* (typically, *reading* from them or *writing* on them, according to the well-known UNIX data protection primitives) *subjects* must secure the authorization of access control systems. Some of these systems are concerned not only about single accesses, but also about propagation of data (*flow control*).

The theory of access control systems has generated extensive literature. Given the importance of the access control function, it is generally agreed that such systems have to be built according to solid *formal methods*. Many such methods have been proposed. The logic of the *models* underlying such systems has often been specified in formal terms, and proofs have been provided of their soundness.

However, there is no agreement on what kind of logic methods can be used in order to prove properties of these models. Different authors have used different concepts and methods, usually tailored to the specific model they were discussing. In standard textbooks [5], the various models are presented with different formalisms. The result is conceptual fragmentation in the area, which makes it difficult to isolate the concepts and to combine solutions.

In this paper, we identify some basic access control and data flow control concepts that can be used for characterizing the invariant properties of abstract models and for providing their proofs. The concepts are intuitive and are shown to be general enough to deal with several of the classical models of access control. In order to keep proofs simple and the paper within page limits, these models have been reduced to their minimal characteristics and adapted to our method. Also, proofs are sketchier or omitted for the more complex models.

We concentrate on proving properties of confidentiality and integrity, for which we propose our own formal definitions. According to an often cited source [18], *confidentiality* is related to disclosure of information, while *integrity* is related to modification of information. In our formal method, confidentiality defines what subjects can know, and integrity defines what objects can contain. We are aware of the fact that these two terms are often used with more complex meanings. Our definitions arise from our concepts and do not attempt to fully capture these meanings.

After a brief literature review in Section 2, the main ideas of this paper are presented in Section 3. The following sections present case studies to show the range of applicability of the method and can be read independently, with the exception mentioned in Section 11. An easy application is coalitions, Section 4. Upward multilayer models, with their counterpart, downward multilayer models, are presented in Section 5 and 6. Section 7 presents a model for categories or domain powersets as we call them. If incompatibilities between domains are defined, then we obtain a sort of static Chinese Wall model, presented in Section 8. A very simplified RBAC model is presented in Section 9. In Sections 10 and 11, we deal with the dynamic models of High Water Mark and Chinese Wall. Section 12 presents prospectives of research.

A note about numbering and references for results: since each section after Section 3 presents a separate example, we have not used a global numbering scheme for our results.

2 Literature review

Over the years, there have been many contributions on logic formalisms and formal methods for access control models. Most of these contributions addressed specific models and applications. Others were more general but went in directions that are orthogonal to ours. Abadi has written many papers on this general subject, but addressing subjects such as authentication, cryptography, security protocols and languages for programming security policies. Survey paper [1] can be used as introduction to his work. Bertino et al. [4] developed a logical framework "to evaluate and compare the expressive power of access control models", which is not the purpose of our paper. Similar goals inspired the work of Habib, Jaume and Morisset [9,12]. Barker [2,3] has developed a general logic framework for the modeling of access control, but his main focus were policy systems, rather than reasoning about access control and flow control properties of models as in our paper. Crampton et al. [6] presented a logic for automated tools "to assist in reasoning about the effects of an access control mechanism on the computations performed by an operating system".

Cuppens and Demolombe [7] developed a modal logic framework to specify confidentiality policies. They addressed indirect channels such as deductive and abductive ones.

Although these papers addressed different problems from ours, there are several possibilities of combining research approaches, which should be the goal of future research.

The access control models we discuss are well-known textbook models and most of them are presented in [5,8]. We refer to these sources for citations and background information. As mentioned, the usual definitions are adapted to our method.

3 Basic concepts for our logical method

Our method uses seven basic concepts:

- Three entities: Subjects, Objects, Data Variables. The objects model files or databases, and data variables stand for data values contained in the objects.
- Two relationships between subjects or objects and data variables, respectively CanKnow and CanStore
- Two relationships that express access control constraints or rules between subjects and objects: CanRead, CanWrite.

We will see that this is enough to define several basic access control models and prove some of their main properties. Other more complex models may require additional concepts. We will see that the High Water Mark and Chinese Wall models will require the notions of state and state transitions caused by read and write operations.

We use the letters S, O and x with various primes or numerals to denote variables over respectively subjects, objects and data items (indexes will be used later to express states). We also use the following abbreviations:

CK for CanKnow, CS for CanStore, CR for CanRead, CW for CanWrite

A simple example will introduce the idea of our method. Consider a system with two subjects S1, S2 and two objects, O1 and O2. Suppose that we have the following access control relationships:

CR(S1,O1), CW(S1,O2), CR(S2,O2).

Suppose also that CS(O1,x) is given unconditionally, so we know that x is in O1. Then S1 can read x from O1 and then store it in O2. S2 can then read x from O2. We can conclude that CS(O2,x) and CK(S2,x).

In other words, subjects can know values by reading them from the data objects they are authorized to read, and objects can store values that are written on them by subjects authorized to do so. In order for something to be known or stored at all in a system we must also assume that some subjects can know something or some objects can store something at some point and this will be expressed by *unconditional relationships*.

Security research knows many other methods for passing information. Covert or indirect channels were first discussed in [13], and other possibilities were identified afterwards. A reference that uses a logical point of view is [7]. Our method does not consider explicitly these other methods, and so does not attempt to cover information flow in its generic meaning but is open to the addition of other entities and rules.

Table 1 defines the deduction rules for our method. The rule for CK expresses the fact that if there exists an object that can store a value and a subject can read that object, then the subject can know the value. The rule for CS deals with storing and it is similar. The rules of Table 1will be used throughout this paper, although for readability we will use their intuitive meaning in place of their logic formulation. However a translation into symbols should always be possible.

- *Unconditional relationships* are expressed in the form: CK(S,x) or CS(O,x)
- The *inference rule for CK* is:
  $\exists O\ (CS(O,x) \wedge CR(S,O)) \rightarrow CK(S,x)$
  (if O can store x and S can read from O, then S can know x)
- The *inference rule for CS* is:
  $\exists S\ (CK(S,x) \wedge CW(S,O)) \rightarrow CS(O,x)$
  (if S can know x and S can write on O, then O can store x)
- All CS or CK relationships must be true either unconditionally or by one of the two inference rules above.

Table 1. Deductive system

We also use the auxiliary functions CSS and CKS, as follows:

For any S, *CanKnowSet(S)* or *CKS(S)* is the set of data variables x for which CK(S,x) is true: CKS(S) $=_{def}$ {x | CK(S,x)=true}

For any O, *CanStoreSet(O)* or *CSS(O)* is the set of data variables x for which CS(O,x) is true: CSS(S) $=_{def}$ {x | CS(S,x)=true}.

We will use both the notation based on CS and CK binary predicates and the notation based on CKS and CSS sets. These two notations are equivalent, e.g. the deductive rules of Table 1 could be expressed in terms of sets, but in intuitive terms one notation is sometimes more clear than the other. E.g. it may be easier to say: S can know x rather than saying: x is in the set that can be known by S. On the other hand it is easier to say that S can know everything that S' can rather than saying that for all x, etc. The set notation will be used in order to define the confidentiality and integrity properties.

Concerning logical notation, in order to simplify the expressions, we omit universal quantifiers. Variables that are not existentially quantified should be assumed to be universally quantified outside the expression. This implies that several free occurrences of the same variable in an expression refer to the same entity. This is a common convention in algebra.

A key element of our logic method is the use of *variable labelling*, e.g. in the next section we use the notation *O:D* to denote object O belonging to domain D. These labels can be thought of as types, since they regulate the use of read and write operations for the variables that are associated with them, by means of CR and CW predicates. However we will not make use of type theory concepts. Our use of labels is inspired by [15] but we are not attempting in this paper to establish a relationship between our method and the method of [15].

We should clarify the role of these labels in implementations of the models described by our method. If a label is used in the access control rules of a model, then it should be available in the implementation, e.g. for the model of the next section the implementation should have available the labels indicating the domains of subjects S or objects O. Otherwise, the label is there only for our proofs, to show the provenance of entities, e.g. in the next section x:D means that data variable x belongs to domain D but an implementation of the model does not need to know this, since the access control rules do not use it. There are security models in which the labels of data variables should be available to the implementation, such is the one of [15].

To lighten the notation we also write variables without labels. This means: any possible label, according to the context. For example in Table 1, x, S and O have no labels which means that the rules apply for any labels, still subject to the rule that if there are several free occurrences of the same variable in an expression, they refer to the same entity and so all such occurrences are understood to have the same label.

4 Coalitions

The study of coalitions is a first application of our method. It is very simple, but introduces a concept that is useful in practice and provides a good introduction to our method. The main properties of coalitions will hold, with appropriate changes, for other models that will be discussed later.

Coalitions model collaborating organizations that share data. A *coalition* is a set of *domains* that are mutually compatible. Data, subjects and objects belong to specific domains but subjects can read or write from or to other objects of their coalition. So an object will store values only from its own coalition. For example, in a coalition of banks, each bank has its own customers, but all banks can read from each other's databases and so can know and store all the data of all the members of the coalition.

Coalitions in their elementary form presented here are not mentioned in the literature (however note that if the conflict relationship to be mentioned in Section 8 is transitive, its complement, to be denoted there as ~ is a coalition [14]). Some papers have considered information sharing schemes with relation to Role-Based Access Control. Other papers study organizational and collaborative issues related to coalitions, spontaneous coalitions, coalitions in social networks, etc., issues which are beyond the scope of this paper. We give coalitions their place in our catalogue of basic models of access control.

To formalize the notion of coalitions, we use the following notation and conventions:

- S:D, O:D, x:D are variables for subjects, objects and values belonging to domain D
- For each D, we assume that there is at least one S:D, one O:D and one x:D
- $D \approx D =_{def}$ D and D' belong to the *same coalition*: this an equivalence relation that is supposed to be known for a given system.

We assume that for each data variable x:D we have at least one of the following two:

- an unconditional relationship of the form CK(S:D, x:D) stating that subject S:D can know x:D

- an unconditional relationship of the form $CS(O{:}D, x{:}D)$ stating that object O:D can store x:D

The access control rules and the confidentiality and integrity requirements for coalitions are given in Table 2:

- (Read rule) $CR(S{:}D, O{:}D') \leftrightarrow D \approx D'$ (a subject can read from an object iff they are in the same coalition)
- (Write rule) $CW(S{:}D, O{:}D') \leftrightarrow D \approx D'$ (a subject can write on an object iff they are in the same coalition)
- (Confidentiality property): $x{:}D \in CKS(S{:}D') \leftrightarrow D \approx D'$ (a value belonging to one domain can be known in another domain iff the two domains are members of the same coalition).
- (Integrity property): $x{:}D \in CSS(O{:}D') \leftrightarrow D \approx D'$ (a value belonging to one domain can be stored in another domain iff the two domains are members of the same coalition).

Table 2. Access control rules and properties for coalitions

The contents of Table 2 can be taken in one of two ways: one could start from the access control rules and prove that the confidentiality and integrity properties are satisfied by the rules. Or one could start from the desired properties and derive the rules. In this paper, we take the first avenue and we leave the second for future research. We have the following results, note that several properties are independent of the label of x.

Property 1 (Pervasiveness of data values in a domain): $CK(S{:}D, x) \leftrightarrow CS(O{:}D, x)$

Proof: Any S:D can write on any O:D and any S:D can read from any O:D.□

The following properties extend the pervasiveness to all domains in a coalition

Property 2: $D \approx D' \rightarrow (CK(S{:}D, x) \leftrightarrow CK(S'{:}D', x))$

Property 3: $D \approx D' \rightarrow (CS(O{:}D, x) \leftrightarrow CS(O'{:}D', x))$

Property 4: $D \approx D' \rightarrow (CK(S{:}D, x) \leftrightarrow CS(O'{:}D', x))$

Proofs: They are similar for all three. If a subject in a coalition can know any value, then it can write the value in an object in the same coalition, from which it can be read by another subject in the coalition, etc.□

Property 5 (Main coalition property): $(CK(S{:}D, x{:}D') \vee CS(O{:}D, x{:}D')) \leftrightarrow D \approx D'$

Proof for →: This property can be true unconditionally and then D=D'. Other CK or CS relationships can be inferred from the unconditional ones only by using CK or CS inference rules. The access control rules ensure that D≈D' at each inference, and ≈ is transitive.

Proof for ←: Both consequences are simultaneously true, and the proofs are similar. Let us prove only $D \approx D' \rightarrow CK(S{:}D, x{:}D')$. By the unconditional relationships, there must exist a subject S:D' that knows x:D' or an object O:D' that stores it. In the first case, x:D' can be written on a O:D'. Since D≈D', S:D can read from it. In the second case, no write is necessary and S:D can read directly from O:D'.□

The following corollary is another way to write the confidentiality and integrity properties of Table 2.

Corollary: $CK(S{:}D, x{:}D') \leftrightarrow D \approx D'$ and $CS(O{:}D, x{:}D') \leftrightarrow D \approx D'$

In view of the previous results, we can strengthen the three pervasiveness properties to ‘if and only if’. Let us take the first one, and the other two follow in similar ways:

Property 2’: (CK(S:D, x) $\leftrightarrow$ CK(S:D’, x)) $\leftrightarrow$ D$\approx$D’

Proof: the $\leftarrow$ part was Property 2, now for (CK(S:D, x) $\leftrightarrow$ CK(S:D’, x)) $\rightarrow$ D$\approx$D’. This means proving: (CK(S:D, x:D’’) $\leftrightarrow$ CK(S:D’, x:D’’)) $\rightarrow$D$\approx$D’. By the Corollary, D$\approx$D’’ and D’$\approx$D’’ so D$\approx$D’.□

This model can be varied in various ways, which we propose for further study. For example, subjects can read only from own domain, but write anywhere in the coalition, or: subjects can read from anywhere in the coalition, but write only in own domain, or yet: subjects in different domains can have different read and write properties, e.g. we can have ‘master’ and ‘slave’ domains. This last variation brings us to the models considered in the next section.

5 Upward multi-level models (UML)

The *upward multi-level models* (UML) are characterized by partially ordered sets of security levels. On the low (high) end of the ordering we find the low (high) security classifications. These models are characterized by the requirement: *information can only move upward*, i.e. from lower to higher security classifications [18]. The well-known Bell-La Padula model (BLP henceforth) belongs to this category. The main simplification we make here is that a single set of levels is used for both security clearance and security classification. So if TopSecret > Secret, subject Alice:TopSecret can read from but not write on object MedFile:Secret.

Furthermore:

- S:L, O:L, x:L are variables for subjects, objects and data belonging to level L
- For each level L, we assume that there is at least one S:L, one O:L and one x:L

This assumption allows us to avoid having to consider some limit cases, such as levels among which no information could flow, levels holding no information of their own, etc. Without it the essential results would remain valid, but with more complicated formulation and proofs. However we don’t assume that the number of levels is finite and we don’t require them to form a lattice.

We assume that for each x:L we have at least one of the following two:

- an unconditional relationship of the form CK(S:L, x:L) stating that a subject S:L can know x:L
- an unconditional relationship of the form CS(O:L, x:L) stating that an object O:L can store x:L

The access control rules and the confidentiality and integrity properties for UML are given in Table 3. The latter state that a value of a certain security level cannot be known or stored at a lower security level, which is another way of saying that information can only move up.

- $CR(S{:}L, O{:}L') \leftrightarrow L \geq L'$ (Subjects can read only at the same level or lower)
- $CW(S{:}L, O{:}L') \leftrightarrow L' \geq L$ (Subjects can write only at the same level or higher)
- (Confidentiality) : $x{:}L \in CKS(S{:}L') \leftrightarrow L \leq L'$
- (Integrity) : $x{:}L \in CSS(O{:}L') \leftrightarrow L \leq L'$

Table 3. Access control rules and properties for Upward Multi-level models

We have the following results, which should be obvious. They state the pervasiveness of data values in a level and the upward monotonicity of knowledge. They are independent of the label of x.

Property 1: $CK(S{:}L, x) \leftrightarrow CS(O{:}L, x)$

Property 2: $L \geq L' \rightarrow (CK(S'{:}L', x) \rightarrow CK(S{:}L, x))$

Property 3: $L \geq L' \rightarrow (CS(O'{:}L', x) \rightarrow CS(O{:}L, x))$

Property 4: $L \geq L' \rightarrow (CK(S'{:}L', x) \rightarrow CS(O{:}L, x))$

Proof of Property 2: By Property 1, an O':L' can store x. By the CR rule, S:L can read it and know it. □

Property 5 (Main UML property):

$$(CK(S{:}L, x{:}L') \vee CS(O{:}L, x{:}L')) \leftrightarrow L \geq L'$$

Proof for →: This property can be true unconditionally and then L=L'. Otherwise, it can be true only by CK and CR inference rules. But then L≥L' must be true at each inference by the access control rules and then the result follows by transitivity.

Proof for ←: By the unconditional relationships, $CK(S'{:}L', x{:}L') \vee CS(O'{:}L', x{:}L')$ for some S:L' or O:L'. Since L≥L', $(CK(S{:}L, x{:}L') \vee CS(O{:}L, x{:}L'))$ can be derived by repeated use of CK or CS inference rules. We have assumed that there are subjects and objects at every level, so the data can move up.□

Property 6 (Strong monotonicity): $(CK(S'{:}L',x) \rightarrow CK(S{:}L,x)) \leftrightarrow L \geq L'$

Proof. The ← was proven earlier and the → can be proven by contradiction. $(CK(S'{:}L',x) \rightarrow CK(S{:}L,x)) \wedge L<L'$ is impossible since we have assumed that at each level, such as L', there is a variable x:L' that cannot be read from level L. □

Property 7 and **Property 8** are corresponding 'iff' strengthenings of Properties 3 and 4 and can be proven in similar ways.

We can express the same properties in terms of sets, with the *proper inclusion* operator ⊂ :

Property 6': $(CKS(S{:}L) \subset CKS(S'{:}L')) \leftrightarrow L<L'$

Property 7': $(CSS(O{:}L) \subset CSS(O'{:}L')) \leftrightarrow L<L'$

Property 8': $(CKS(S{:}L) \subset CSS(O'{:}L')) \leftrightarrow L<L'$

The confidentiality and integrity properties of Table 3 follow as corollaries.

6 Downward multi-level models (DML)

We have just seen that UML models allow values to migrate to higher levels. Therefore, higher levels are not protected against information coming from lower levels. This is usually expressed by saying that UML leads to high confidentiality but low integrity. To protect the integrity of the upper layers, the Biba model was invented, where *information can only move downward* [18]. The duality of *downward multi-level models* (DML) with respect to UML is well-known, and we leave the proof of the corresponding properties as an easy exercise.

We propose as an interesting case study a ‘maximum integrity model’ that we have found in a company. Some important documents of this company are placed in a security level that can be read by all, but cannot be written on. Furthermore, information belonging to this level cannot be stored anywhere else. In this way, these documents do not risk becoming corrupted. Note that in this model we don’t have the symmetry between confidentiality and integrity that characterizes the previous models: all levels can know but only one can store this information. Unlike the previous models, the labels of the data variables seem to be important in implementations of this model.

7 Domain powerset models (DP)

Domain powerset models (DP) are often called *category* or *compartment* models and are combined with the UML model in the usual presentation of the BLP model. However they deserve a place of their own because they can be combined with other models than UML.

Let us assume that in an organization there are three domains (or categories or compartments) of data, for example: NUC, EUR, US. Some subjects can be allowed to access only data classified NUC, others only data classified EUR, other yet only data classified US. We label them {NUC},{EUR},{US}. But then there may be subjects allowed to access data classified NUC or EUR, they will be labeled {NUC, EUR} and so upward in the powerset lattice until we find the label {NUC, EUR, US}. There may also be subjects not allowed to access anything, labeled {}. See for this [5]. We extend this labeling method to objects, to specify what are the domains of the values that can be stored in them.

Following our earlier conventions, we will continue using D with primes for variables for domain names. Labels Δ, Δ’ ... will denote sets of domains. A subject S:Δ has clearance to know only data variables in the domains in Δ. An object O:Δ can store data belonging to any of the domains in Δ. We also assume that there is at least one object, subject and data variable in each of the domains.

For each variable x:D we have at least one of the following two:

- An unconditional relationship of the form CK(S:{D}, x:D)
- An unconditional relationship of the form CS(O:{D}, x:D)

The access control rules and the properties for this model are given in Table 4 and their proofs follow by methods similar to those we have seen.

- CR(S:Δ, O:Δ’) ↔ Δ’⊆Δ (a subject can read from an object iff the object can contain only data variables that the subject can know)
- CW(S:Δ, O:Δ’) ↔ Δ⊆Δ’ (a subject can write on an object iff the subject can know only data variables that the object can contain).
- (Confidentiality): x:D∈CKS(S:Δ) ↔ D∈Δ
- (Integrity) x:D∈CSS(O:Δ) ↔ D∈Δ

Table 4. Access control rules and properties for Domain Powerset Models

8 Domain powersets with conflicts (DPC)

The existence of conflicts between domains motivates this model, which is a ‘static’ version of the Chinese Wall (ChW) model (see Section 11). Starting from the domain powerset model (DP), we define conflicts between domains, and then we stipulate that *subjects (objects) may not know (store) values belonging to mutually conflicting domains*: this phrase essentially expresses the confidentiality and integrity properties of this model. This means that labels containing conflicting domains are not allowed in DPC, neither for subjects, nor for objects. Note that the resulting label inclusion structure is no longer a lattice. References [17,18] present a lattice construction for ChW. This would be possible for our model as well, if we agree that there could be unassignable labels, or labels that are assignable only to administrative entities.

Rather than using a conflict relationship we choose to use its complement, a compatibility relationship, which we assume to be reflexive and symmetric, but not necessarily transitive as in coalitions. For domains D and D’ we write D∼D’ if the two domains are compatible. We suppose that this relationship is known in a given organizational context. We still use labels Δ as for the DP model, but with the condition that for D,D’ ∈ Δ, D∼D’.

Because of this fact, the CR and CW relationships for the DPC model can be the same as those of the DP model in Table 4, and the confidentiality and integrity properties follow.

As an example, consider a system with two subjects, Alice and Bob, and three domains, Bank1, Bank2 and Oil. The domains are all pairwise compatible, with the exception of Bank1 and Bank2. So allowed labels are: {}, {Bank1}, {Bank2}, {Oil}, {Bank1, Oil}, {Bank2, Oil} and forbidden labels are: {Bank1, Bank2}, {Bank1, Bank2, Oil}. A possible label assignment is: Alice: {Bank1, Oil}; Bob: {Oil}; Bank1: {Bank1,Oil}; Bank2: {Bank2, Oil};Oil: {Oil}. We’ll leave it as an exercise for the reader to determine the resulting CR and CW relationships.

An issue in this model is that a conservative label assignment could assign to all subjects only empty labels. This of course would come into conflict with “need to know” and availability requirements, and is a reason for the model of Section 11.

9 Role-based access control (RBAC)

In order to present the essential idea in one page, we will consider here a drastically simplified RBAC with only subjects, objects and data variables. We will take the concept of RBAC's *role* to correspond to our concept of *subject* and subjects-roles will be called R1, R2 etc.

Subjects-roles will be labeled with their sets of permissions P, each of which will be of the form (CR,O) or (CW,O). Normally objects are not labeled in RBAC. The access control rules for RBAC are given in Table 5.

- CR(R:P, O) ↔ CR(R,O)∈P
- CW(R:P,O) ↔ CW(R,O)∈P

Table 5. Access control rules for RBAC

Example:

- R1:{(CR,O1),(CW,O2)}
- R2: {(CR,O1),(CR,O2)}
- R3: {(CR,O1), (CR,O2),(CW,O2),(CW,O3)}
- R4: {(CR,O3)}

We assume unconditional relationships of the form CK(Oi, xi:Oi) for each Oi. By application of the inference rules of Table 1 and the access control rules of Table 5 in the example above the following can be derived:

- CKS(R1)={x1:O1};
- CKS(R2)=CKS (R3)={x1:O1, x2:O2};
- CKS(R4)={x1:O1, x2:O2, x3:O3}
- CSS(O1)={x1:O1};
- CSS(O2)={x1:O1, x2:O2};
- CSS(O3)={x1:O1, x2:O2, x3:O3}

Constraints can be defined in RBAC according to the application. Here are two constraints that are violated in the previous example:

- (Confidentiality) No subject-role should be allowed to know both x1:O1 and x2:O2
- (Integrity) No object should be able to store both x1:O1 and x2:O2

A practical case for this integrity constraint could be: no cheque can bear two accountant signatures. This constraint is violated if O2 is a cheque and x1:O1, x2:O2 are accountant signatures.

Other possibilities exist for the application of this method in RBAC, but this should be the subject of another paper.

A more complete presentation requires considering mappings between users, subjects and roles, as well as mappings between roles and sets of permissions. See [16] for a thorough discussion of data flow in RBAC.

10 High water mark (HWM)

HWM is our first example of a dynamic model. In dynamic models, some read or write operations that are not allowed by the access control rules are still possible and lead to state changes, i.e. label changes. Note that our description of this model differs from the usual one, but the idea is related.

In practice, how can we justify such read and write operations? This can vary from model to model. We can think that they are allowed after a request to a system administrator, or as a result of progression in a workflow, or yet as the result of an exception. Some models, such as the Chinese Wall model described below, allow them automatically although they may preclude some other operations later.

The HWM model is a variation of the UML model. We define it as follows (note that the usual definition is somewhat different). Writing down is possible but when this happens, the level of the object which has received the information from higher up is moved up to the level of the subject that has just written on it. Reading up is possible but as soon as a subject does this, its level is moved up correspondingly.

This model can be formalized by a state machine whose transitions are determined by read and write operations. As a consequence of the operations, subjects or objects can change levels so that the confidentiality and integrity properties of UML models can remain true. Data variables don't change labels.

We have an unbounded set of states M_0, M_1 … $L_i(S)$ or $L_i(O)$ are the labels of S or O at state M_i. The labels denote levels as in UML and the set of levels is the same for all states. The notation CKS_i, CSS_i has similar meaning. Each one of these sets is determined by what can be read or stored at state M_i according to the access control rules of the UML model.

Note that although now the notation $S:L_i(S)$ or $O:L_i(O)$ would be formally more appropriate, we will continue to use the notation $S:L_i$ or $O:L_i$ for simplicity. A consequence is the fact that L_i is not the same in $S:L_i$ and $O:L_i$.

For the initial state M_0 and each x:L we have one of the unconditional relationships: $x:L \in CKS_0(S:L_0)$ or $x:L \in CSS_0(O:L_0)$ where $L_0=L$.

R(S,O) and W(S,O) are respectively read and write operations of a subject on an object which cause state transitions. They are not constrained by the access control rules and have no pre-conditions, but have post-conditions.

The post-condition of a $R(S:L_i, O:L'_i)$ operation is:

- For $S:L_{i+1}$, L_{i+1} is the greatest of L_i and L'_i
- However for $O:L'_{i+1}$, $L'_{i+1} = L'_i$ (Upgrade security level of subject, but the object does not change level)

From this we can derive:

- $CKS_{i+1}(S:L_{i+1}) = CKS_i(S:L_i) \cup CSS_i(O:L'_i)$ and $CSS_{i+1}(O:L'_{i+1}) = CSSi(O:L'_i)$ (We add to what subject can know, but the object's knowledge is unchanged)

Correspondingly, the post-condition of a $W(S:L_i, O:L'_i)$ operation is:

- For $O:L'_{i+1}$, L'_{i+1} is the greatest of L_i and L'_i
- However for $S:L_{i+1}$, $L_{i+1}= L_i$

From this we can derive:

- $CKS_{i+1}(O:L'_{i+1}) = CKS_i(S:L_i) \cup CSS_i(O:L'_i)$ and $CSS_{i+1}(S:L_{i+1}) = CSS_i(S:L_i)$

The confidentiality and integrity properties of Table 6 follow for the HWM model, by a simple induction proof. The properties hold for the unconditional relationships at the initial state and the post-conditions for each state transition ensure their invariance.

- $x{:}L \in CKS_i(S{:}L_i) \leftrightarrow L \leq L_i$ (Confidentiality)
  (a variable can be known by a subject at some state iff the classification of the variable does not exceed the classification of the subject at that state)
- $x{:}L \in CSSi(O{:}L_i) \leftrightarrow L \leq L_i$ (Integrity)
  (a variable can be stored in an object at some state iff the classification of the variable does not exceed the classification of the object at that state)

Table 6. Properties for HWM

Note that R(S,O) (or W(S,O)) operations could be executed when the access control conditions CR(S,O) (or CW(S,O)) are true. However then the initial and final states of the transition are identical.

The theory knows also a "low water mark" model in connection with DML models and the same concepts apply, with the appropriate modifications.

11 Chinese Wall (ChW)

The usual definitions of ChW characterize this model in terms of constraints on reading and writing expressed by 'simple' and '*' properties [17,5]. We start instead from our DPC model, with its confidentiality and integrity properties. Unlike the DPC model, in our version of ChW subjects initially know nothing, and objects know only one domain. Subjects (objects) are allowed to acquire new information by read (write) operations and then their labels change, but these operations are allowed only if they lead to allowed labels.

There would be much to say about this model, but we limit ourselves to the minimum necessary to illustrate the application of our method. So we consider only the case where the sets of domains, subjects and objects are fixed.

The understanding of our version of ChW depends on the understanding of the DPC model (Section 8) and the HWM model (Section 10). We start from a set of domains {D1, ... Dn}, with a compatibility relation ~ between domains as in the DPC model. The state model is similar to the one of HWM and similar notation will be used, however here there are both pre-conditions and post-conditions for reading and writing.

Labels Δ_i are subsets of the set of domains. For $S{:}\Delta_0$, $\Delta_0=\{\}$and $CKS_0(S{:}\Delta_0)=\{\}$. For $O{:}\Delta_0$, $\Delta_0=\{Dk\}$ for some k and $CSS(O{:}\Delta_0)$ includes the set of all x:Dk. So at the beginning subjects know nothing, and each object stores information about one domain. At each state, the access control rules are the same as for the DPC (or DP) model and determine the CKS_i, CSS_i sets but reading and writing operations R and W are not constrained by them, they are only constrained by their pre-conditions. In $S{:}\Delta_i$, Δ_i is the set of labels of data variables in $CKS_i(S{:}\Delta_i)$ and similarly for $O{:}\Delta_i$.

We write $\Delta\sim\Delta'$ if for $D\in\Delta$, $D'\in\Delta'$, $D\sim D'$. $\Delta\sim\Delta'$ is the pre-condition for both $R(S{:}\Delta, O{:}\Delta')$ and $W(S{:}\Delta, O{:}\Delta')$. In the case of operation R this means that all the information that S already can know must be compatible with the all the information that O already can store, and similarly in the case of operation W.

The post-condition for $R(S{:}\Delta_i, O{:}\Delta'_i)$ describes the fact that S can now know anything in O:

- For $S{:}\Delta_i$, $\Delta_{i+1} = \Delta_i \cup \Delta'_i$
- For $O{:}\Delta_i$, $\Delta_{i+1} = \Delta_i$

From this we can derive:

- $CKS_{i+1}(S{:}\Delta_{i+1}) = CKS_i(S{:}\Delta_i) \cup CSS_i(O{:}\Delta'_i)$
- $CSS_{i+1}(O{:}\Delta'_{i+1}) = CSS_i(O{:}\Delta_i)$

The post-condition for $W(S{:}\Delta_i, O{:}\Delta'_i)$ describes the fact that O can now store anything that S knows:

- For $O{:}\Delta_i$, $\Delta_{i+1} = \Delta_i \cup \Delta'_i$
- For $S{:}\Delta_i$, $\Delta_{i+1} = \Delta_i$

And from this we can derive:

- $CSS_{i+1}(O{:}\Delta_{i+1}) = CSS_i(O{:}\Delta_i) \cup CSS_i(O{:}\Delta'_i)$
- $CKS_{i+1}(S{:}\Delta'_{i+1}) = CSS_i(S{:}\Delta_i)$

The confidentiality and integrity properties for this model are the same as those for the DPC model, and follow by simple induction on the state transitions.

Note that the CR and CW access control rules for the DPC model imply the pre-conditions for the R and W operations. If these are executed when the access control rules are true, the initial and final states of the transition will be identical.

Following is an example of execution of such a system, similar to the example of Section 8.

We have four objects and four domains. At the beginning each object contains one domain, which has the same name as the object:

Bank1:{Bank1}, Bank2:{Bank2}; Oil:{Oil}; Auto:{Auto}

We suppose that we have only one conflict, between domains Bank1 and Bank2. We have two subjects, Alice and Bob, who know nothing at the beginning, their labels are {}. A possible sequence of state transitions is:

Alice reads from Bank1, now Alice:{Bank1}.

Bob reads from Bank2, now Bob:{Bank2}.

Alice reads from Oil, now Alice:{Bank1,Oil}.

Bob writes on Oil, now Oil:{Oil,Bank2}.

(Henceforth, Alice cannot read from Oil, since it may contain Bank2 data).

Alice writes on Auto, now Auto:{Auto,Bank1,Oil}

(Henceforth, Bob cannot read from Auto, since it may contain Bank1 data)

Each one of the above is a R or W operation that changes the state. After a number of such operations we will reach a state after which no new labels can be generated. But in a real execution after this and between state transitions other read and write operations are possible, which agree with the DPC (or DP) access control rules and therefore do not change the state.

It is clear in this example that it is not possible to reach a state where Alice or Bob can know both x:Bank1 and y:Bank2, and similarly for what objects can store.

Note that according to the original definition of ChW Alice would be prevented to write on Auto. This example shows that the two shortcomings of this definition identified in [17] are not present in our version of ChW. Similar results were shown in [17,19] by using different models.

12 Future research: Variations and combinations of the models

Once a basic common reasoning method has been developed, and once some basic models have been identified, it is possible to start playing with variations and combinations of the models. We have mentioned that by combining the UML model with the DP model we obtain the most commonly referenced variety of the BLP model. It is also known that we can further combine this with the DML models to obtain a model where UML is used for certain types of information (e.g. sales statistics), and DML for others (e.g. directives). Since these models have been invented, many variations for them have been proposed in the literature and it will be interesting to see which ones fit in our framework, and if not what extensions will be necessary.

Aspects of *Discretionary Access Control* (DAC) models can be described by using labels to describe capabilities of subjects or access control lists for objects, where label changes result in state changes.

New models can also be invented. For example, one can think of combinations of the RBAC model with layered models, if subject and object variables are given levels in additions to permissions. Such models can be made dynamic if labels can change by effect of operations, as a consequence of user action or administrative intervention.

References [20,10] propose Attribute Based Access Control (ABAC) as a very general access control model where these new combined models could find their places. Applications of these models can perhaps be found in the Web and in the Cloud, where there is a need for multi-functional access control models [11].

Of course, many concepts can be added to our model: operations different from reading and writing, creation and deletion of subjects and objects, etc.

13 Conclusions

We have presented a new formalized reasoning method for expressing and proving properties of access control and data flow control models. This method is based on simple and intuitive concepts, but has been shown to be usable for a number of classical access control models. The properties proven were already known, but they were originally proven by using different formalisms for each model, if they were proven at all.

Some classical proof methods in this area deal with subject and objects and only implicitly with data. Our method deals explicitly with data, making it possible to prove that data values originating from certain levels or domains cannot cross certain boundaries.

In all the examples we have used essentially the same reasoning method. Each model has access control rules that are defined to ensure the invariance of the model's confidentiality and integrity properties, but this invariance must be proven. Unconditional relationships are specified to populate the model and they are such that the desired properties are true in a base case. Then the fact that the properties are invariant is proven by an induction step showing that, because of the access control rules, each use of an inference rule produces results that preserve the properties. The proofs in this paper were straightforward but they will not necessarily be so for the combined models proposed in Section 12.

The use of our method has allowed us to determine that some of the classical models can be decomposed into more elementary models. Our common logic framework allows these basic models to be varied and combined in many ways, leading to other models that have not been studied in the literature but can be practically useful. Their properties can be proven with our framework.

One interesting question towards which progress can now be made is the following: given certain desired flow control properties, find a labeling principle and a combination of reading and writing authorizations that can be used to achieve it.

Our method can be at the basis of automatic proof methods for more complex models and can be an asset for teaching systematically access control and flow control theory.

Acknowledgment. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada. I would like to thank Frédéric Cuppens and Nora Cuppens-Boulahia for having hosted me during a phase of this project. I would also like to thank my PhD students Sofiene Boulares, Omar Abahmane and Jamal Abd-Ali for many inspiring discussions.

References

1. M. Abadi. Logic in access control. Proc. 18th IEEE Symp. on Logic in Computer Science, (LCS 2003), 228–233.
2. S. Barker. Access control for deductive databases by logic programming, Proc. Intern. Conf. Logic Progr. (ICLP 2002), Springer LNCS 2401, 54–69.
3. S. Barker. The next 700 access control models or a unifying meta-model? Proc. 14th ACM Symp. on Access Control Models and Technologies (SACMAT 2009), 187–196.
4. E. Bertino, B. Catania, E. Ferrari, P. Perlasca. A logical framework for reasoning about access control models. ACM Trans. on Inform. System Security (TISSEC) 6(1), 2003, 71-127.
5. M. Bishop. *Computer Security – Art and Science.* Addison-Wesley, 2003.
6. J. Crampton, G. Loizou, G. O'Shea. A logic of access control. Computer J., 44(2) 2001, 137-149.
7. F. Cuppens, R. Demolombe. A modal logical framework for security policies. Proc.10th Intern. Symp. on Foundations of Intelligent Systems (ISMIS 1997), 579-589.
8. T. Jaeger. *Operating System Security.* Morgan and Claypool, 2008.

9. M. Jaume, C. Morisset. Un cadre sémantique pour le contrôle d'accès. Technique et Science Informatique, 27(8) 2008, 951-976.
10. X. Jin, E. Krishnan, R. Sandhu. A unified access control model covering DAC, MAC, and RBAC. Proc. DBSec 2012, Springer LNCS 7371, 41-55.
11. S. Khamadja, K. Adi, L. Logrippo. Designing flexible access control models for the Cloud. Proc. 6th Intern. Conf. on Security of Inform. and Networks (SIN 2013), 225-232.
12. L. Habib, M. Jaume, C. Morisset. Formal definition and comparison of access control models. Journ. of Inform. Assurance and Security 4, 2009, 372-381.
13. B.W. Lampson. A note on the confinement problem. Comm. ACM, 16(10) 1973, 613-615.
14. T.Y. Lin. Placing the Chinese Walls on the Boundary of Conflicts - Analysis of Symmetric Binary Relations. Proc. 26th Ann. Intern. Comp. Softw. and Appl. Conf. (COMPSAC 2002), 966-971.
15. A.C. Myers, B. Liskov. Protecting privacy using the decentralized label model. ACM Trans. on Softw. Eng. and Methodology (TOSEM), 9(4) 2000, 410-442.
16. S. Osborn. Information Flow Analysis of an RBAC System. Proc. 7th ACM Symp. on Access Control Models and Technologies (SACMAT 2002), 163-168.
17. R.S. Sandhu. Lattice-based enforcement of Chinese Wall. Computer and Security, 11(8), 1992, 753-763.
18. R.S. Sandhu. Lattice-based access control models. Computer 26(11), 1993, 9-19.
19. A. Sharifi, M.V. Tripunitara. Least-restrictive enforcement of the Chinese Wall security policy. Proc.18th ACM Symp. on Access Control Models and Technologies (SACMAT 2013), 61-72.
20. L. Wang, D. Wijsekera, S. Jajodia. A logic-based framework for Attribute-Based Access Control. Proc. 2004 ACM workshop on Formal methods in security engineering (FMSE 2004), 45-55.