FSL — Fibred Security Language

Valerio Genovese[1], Dov M. Gabbay[2], Guido Boella[1], Leon van der Torre[3]

[1] Dipartimento di Informatica. Università di Torino - IT.
E-mail: guido@di.unito.it; valerio.click@gmail.com
[2] Dept. Computer Science, King's College London - UK.
E-mail: dov.gabbay@kcl.ac.uk
[3] Computer Science and Communications, University of Luxembourg, Luxembourg.
E-mail: leon.vandertorre@uni.lu

Abstract. We develop a fibred security language capable to express statements of the form

$$\{x\}\varphi(x) \textbf{ says } \psi$$

where $\{x\}\varphi(x)$ is the set of all x that satisfy φ and ψ is any formula. φ and ψ may share several free variables. For example, we can express the following: "A member m of the Program Committee can not accept a paper P_1 in which one of its authors says that he has published a paper with him after 2007"

$$\neg(\{m\}[PC(m) \wedge \{y\}author\_of(y, P_1) \textbf{ says } \exists p(paper(p) \wedge author\_of(m,p) \wedge author\_of(y,p) \wedge year(p) \geq 2007)] \textbf{ says } accept(P_1))$$

1 Introduction

Access control is a pervasive issue in security: it consists in determining whether the principal (a machine, user, program) that issues a request to access a resource should be trusted on its request, i.e., if it is authorized. Authorization can be based in the simplest case on access control lists (ACL) associated with resources or with capabilities held by principals, but it may be complicated by, for instance, membership of groups, roles and delegation. Thus, logics for access control are often used to express policies and to enable reasoning about principals and their requests, and other general statements.

In many cases first-order/propositional logic suffices, but it does not in the case of distributed policies and delegation, e.g., "administrator says that Alice can be trusted when she says to delete $file_1$": Alice speaks for the administrator concerning the deletion of $file_1$, thus she should be trusted as much as the administrator.

In this paper we present a Fibred Security Language (FSL) for access control in distributed systems. Fibring is a general methodology due to Gabbay [1] that aims to combine logics.

Dagstuhl Seminar Proceedings 09121
Normative Multi-Agent Systems
http://drops.dagstuhl.de/opus/volltexte/2009/1908

Suppose we have two different logics $\mathcal{C}$ and $\mathcal{D}$ with languages $\mathbb{L}_{\mathcal{C}}$, $\mathbb{L}_{\mathcal{D}}$ and semantics $S_{\mathcal{C}}$, $S_{\mathcal{D}}$ respectively. Intuitively, the fibring process consists in defining a combined language $\mathbb{L} \supset \mathbb{L}_{\mathcal{C}} \cup \mathbb{L}_{\mathcal{D}}$ together with a new semantics S in which we can evaluate formulas of both $\mathcal{C}$ and $\mathcal{D}$.

From a semantical point of view, logics for distributed access control rely on one of the following approaches

- Operational Semantics [2].
- Declarative Semantics [3,4].
- Classical/Intuitionistic Modal logic [5,6,7,8].

Each view has positive and negative aspects.

Operational Semantics, if rules are wisely crafted, could be extremely clear but very often tractability must be sacrificed for simplicity. SecPAL, for instance, has an extremely clear semantics expressed with just three rules, but in practice they are awkward to employ in evaluating formulas. To overcome this difficulty queries in [2] are evaluated exploiting Datalog that has a stable model semantics which is not clearly related with the rules of the operational semantics.

Logics that rely on declarative semantics have a clearly specified notion of proof of compliance which is strictly based on the framework in which the reasoning is carried out. PROLOG and Datalog seems to be the most used solutions to obtain answer sets from a database of distributed policies. The negative aspect is that using declarative approaches it could be extremely difficult to have a formal "meaning" for every set of policies and credentials, so that one can compute this meaning and inspect whether it is the same as the policy author's intention.

Modal logic have been employed by Abadi [6] to model logics for access control, in this view a logic can be studied through its axiomatization or on the basis of its semantics analyzing how to link models with formulas. One major advantage is that with a clear bound between syntax and semantics the proof of compliance procedure is based on well-understood, formal foundation. A mayor loss is that it could be extremely difficult to compose different logics within a common framework if we do not rely on fibring.

Every approach has some positive aspects that should not be left out in modelling a logic for distributed access control. With FSL we propose a general language to compose (fibring) existing logics on the basis of their semantics, in particular Section 4.2 is devoted to introduce an authorization logic called predicate FSL in which we fibre intuitionistic logic with multimodal logic. Future papers will be devoted to extend and compose existing access control logics (see Section 6).

In predicate FSL we have formulas of the kind

$$\{x\}\varphi(x) \textbf{ says } \psi \tag{1}$$

where $\{x\}\varphi(x)$ represents the group composed by all the principals[4] that satisfy $\varphi(x)$ and ψ is a general formula. We see the **says** as a modality to express that a certain principal supports some statement (see Section 2).

In this view, Formula 1 becomes

$$\Box_{\{x\}\varphi(x)}\psi \tag{2}$$

In which ψ is the statement that the extension of $\varphi(x)$ *as a group* of individuals supports; note also that the modality is indexed by principals. Up to authors knowledge, existing approaches that employ the **says** operator do not offer the possibility to have a first-order formula specifying the principals.

This view on access control logics offers a wide range of expressiveness in defining policies and freedom in crafting logics. In fact we can let $\varphi(x)$ and ψ belong to two different languages $\mathbb{L}_p$ and $\mathbb{L}_e$ as language of principals and security expressions respectively which refers to two different systems (semantics).

For instance we can think of formulas in $\mathbb{L}_p$ be SQL queries and formulas in $\mathbb{L}_e$ be Delegation Logic [3] expressions.

The main problem is to formally specify how to evaluate expressions like 2 and this is the main role of the fibring methodology [1] which, depending on the chosen languages (and systems), must be carefully defined in order to have a combined logic which is coherent and does not collapse.

In this paper, in order to show the full expressiveness of our approach, we decide to make $\mathbb{L}_p = \mathbb{L}_e = \mathbb{L}$, where $\mathbb{L}$ is a classical first order language, whereas the relying system S is intuitionistic modal logic; this is predicate FSL. This approach offers us to iterate the **says** modality and to have extremely complex formulas in which free variables are shared between different levels of nesting of the $\Box$ (see Section 3.1 for examples).

Throughout the paper we will show how with predicate FSL is possible to give answers to the following questions:

1. How to define a general semantic model in order to extend existing security languages?
2. How to make a principal speak for another principal on all formulas without resorting second order languages?
3. How to have groups of principals supporting a sentence expressed by a first-order formula with free variables?
4. How to express chain of delegation by means of the **says** modality and how to constrain delegation depth?
5. How to express separation of duties in a clear and compact way?
6. How to deal with roles in distributed access control?

The paper is structured as follows. First, in Section 2 we discuss which properties of the **says** operator are desirable in logic and which are not, highlighting the dependencies among them in different logics. Second, in Section 3 we consider how to extend the authorization logic on the side of the principals which

[4] Example of principals are: Users, machines, channels, conjunction of principals, groups . . . [6]

can assert **says** statements. Then, we present the basic *fibred security language FSL* in Section 4 and we extend it to predicate logic in Section 4.2. In Section 5 we give a simple example to show how to employ predicate FSL and Section 6 ends the paper.

2 Properties of access control logics

In this section we first summarize how the **says** operator is used in access control logics, and then we discuss which properties are desired for this operator and which are not, showing the dependencies among the different properties in existing logics.

2.1 Access control logics

The access control logic we propose aims at distributed scenarios. Thus, to express delegation among principals, it is centered, like the access control logic of [5,3], on formulas such as "A **says** s" where A represents a principal, s represents a statement (a request, a delegation of authority, or some other utterance), and **says** is a modality. Note that it is possible to derive that A **says s** even when A does not directly utter **s**. For example, when the principal A is a user and one of its programs includes **s** in a message, then we may have A **says s**, if the program has been delegated by A. In this case, A **says s** means that A has caused **s** to be said, that **s** has been said on A's behalf, or that A supports **s**.

We assume that such assertions are used by a reference monitor in charge of making access control decisions for resources, like o. The reference monitor may have the policy that a particular principal A is authorized to perform $Do(o)$. This policy may be represented by the formula: $(A \textbf{ says } Do(o)) \rightarrow Do(o)$, which expresses that A controls $Do(o)$. Similarly, a request for the operation on o from a principal B may be represented by the formula: B **says** $Do(o)$. The goal of the reference monitor is to prove that these two formulas imply $Do(o)$, and grant access if it succeeds. While proving $Do(o)$ the reference monitor does not need that the principal B controls **s**. Rather it may exploit relations between A and B and some other facts. For example, it may knows that B has been delegated by A, and, thus, that B speaks for A as concerns $Do(o)$, in formulas:

$$(B \textbf{ says } Do(o)) \rightarrow (A \textbf{ says } Do(o))$$

This simple example does not show the subtleties arising from the formalization of the **says** operator, since expressing simple properties like controlling a resource or speaking for another principal may imply less desirable properties, leading to security risks, or even to inconsistent or degenerate logic systems [9].

2.2 Modality axioms

The following are some axioms considered in the literature for the operator **says** , in particular by [9]. We discuss whether they are desirable or not, and which are the relationships among them in different logics, in particular, classical and intuitionistic logic. We write A **says** X as $\Box_A X$. A might be an index U and X ranges over formulas.

Definition 1 (Axiom list).

1. B speaks for A *(notation $B \Rightarrow A$):*

$$\forall X[\Box_B X \to \Box_A X].$$

   Note that here we are quantifying over formulas but if we take it as an axiom schema for the relation between A and B, this will automatically be universally quantified.
   This is the fundamental relation among principals in access control logics. If $B \Rightarrow A$ from the fact that principal B says something means the reference monitor can believe that principal A says the same thing. This relation serves to form chains of responsibility: a program may speak for a user, much like a key may speak for its owner, much like a channel may speak for its remote end-point. In some logics this relation is primitive. The reference monitor's participation is left implicit, as in the all the other axioms.
2. Restricted speaks for

$$\alpha(X) \wedge \Box_B X \to \Box_A X$$

   where $\alpha(X)$ be any formula and X a new variable.
   Restriction of "speaks for" is similar to the one [10] introduces. In particular, if $\alpha(X) = \varphi \to X$, then the above formula would refer to B speaks for A on all consequences of φ [8].
   Other kinds of restrictions can refer to variables occurring in X. We consider such kind of constraints in Section 3.
3. A controls X

$$\Box_A X \to X$$

   This axiom is used in other axioms below.
4. Hand-off axiom

$$\Box_A \forall X[\Box_B X \to \Box_A X] \to \forall X[\Box_B X \to \Box_A X]$$

   or more briefly:

$$\Box_A(B \Rightarrow A) \to (B \Rightarrow A)$$

   Hand-off states that whenever A says that B speaks for A, then B does indeed speak for A. This axiom allows every principal to decide which principals speak on its behalf, since it controls the delegation to other principals.
   Sometimes this axiom follows from logic rules as in [9], sometimes it is assumed as an axiom. Note that the general axiom is too powerful, and thus risky for security: for example when A represents a group: if A **controls** *$(B \Rightarrow A)$ then any member of A can add members to A. Thus, for instance, [6] does not adopt this axiom.*

5. Generalised Hand-off
Since A **controls** X *is defined as* $\Box_A X \to X$*.*
Then
$$\forall XY(A \textbf{ controls } (X \to \Box_A Y))$$
or explicitly
$$\Box_A(X \to \Box_A Y) \to (X \to \Box_A Y)$$
For $X = \Box_B Y$*, we get hand-off:*
$$\Box_A(\Box_B Y \to \Box_A Y) \to (\Box_B Y \to \Box_A Y)$$
Generalised Hand-off is equivalent to Bind (see item 12 below). It follows from logic rules in [9].
6. Dual of Hand-off
$$\Box_A(A \Rightarrow B) \to (A \Rightarrow B)$$
This is implied by Unit in CDD [9], where it is equivalent to Unit axiom if there is a truth telling principal.
7. Least privilege
$$(X \to Y) \to (\Box_A X \to \Box_A Y)$$
"Every program and every user of the system should operate using the least set of privileges necessary to complete the job" [11].
8. Ordinary modal axioms
   – Closure under consequence
$$\Box_A X \wedge \Box_A(X \to Y) \to \Box_A Y$$
   – Necessitation
$$\vdash X \textit{ implies } \vdash \Box_A X$$
9. Axiom C4
$$\Box_A \Box_A X \to \Box_A X$$
10. Escalation
$$\Box_A X \to X \vee \Box_A \bot$$
Escalation is not considered as a desirable property. Thus we must be careful that it does not follow from other properties (like from Unit or Bind in classical logics). It amounts to "if A **says** s *then* s *or*A **says** $false$*": from* A **says** s *may follow a statement "much falser" than* s*. As an example of its riskiness, consider that from* (A **controls** s) $\wedge$ (B **controls** s) *it allows to infer that if* A **says** B **says** s *then* s *follows. If the logic is not able to avoid escalation, the only cumbersome solution is to make* A *avoid saying that* B *says* s *unless he really wishes to say* s*.*
Unit and Bind together do not imply Escalation in CDD [9], while Escalation implies Bind. In classical logic, Unit implies Escalation while Escalation does not imply Bind.

11. Unit

$$X \to \Box_A X$$

Unit is stronger than the necessitation rule. In classical logic, adopting Unit implies that each principal either always says the truth or it says false: $(A \to B) \vee (B \to A)$*. In the first case* A *speaks for any other principal, in the latter any other speaks for* A*. The policies described by this kind of systems are too manicheist.*

12. Bind

$$(X \to \Box_A Y) \wedge \Box_A X \to \Box_A Y$$

Abadi [9] provides an example of discussion about the implications of the different axioms of access control logics.

According to [9] in classical logic, Bind is equivalent to escalation and Unit implies Escalation. Intermediate systems requiring C4 do not lead to escalation, but they are not sufficient for modelling delegation.

To solve this problem Abadi in [9] introduces CDD, a second-order propositional intuitionistic logic; in Section 4.2 we present predicate FSL which extends CDD expressiveness without using a second-order language.

3 Reasoning about principals

In the previous section we considered the properties of the **says** operator keeping the principal indexing the modality as a propositional atom[5]. In this section we make a further step towards predicate FSL taking into account how to express the key properties of access control policies in the proposed language.

3.1 FSL: An extended logic of principals

The logic we propose uses a construct which allows to build principals using general logic formulas: $\{x\}\varphi(x)$ **says** ψ. In this section we will show how we can exploit it. Note that $\varphi(x)$ and ψ can share variables and φ may include occurrences of the **says** operator. Notice that x can occur in ϕ but then this occurrence is not related to the x in $\{x\}\varphi(x)$. The formula $\{x\}\varphi(x)$ is used to select the set of principals making the assertion **says**.

To select a single principal whose name is A we do:

$$\{x\}(x = A) \textbf{ says s}$$

We write A **says s** for $\{x\}(x = A)$ **says s**, where A is an individual principal.

The following formula means that all $user$s together ask to delete $file_1$:

$$\{x\}user(x) \textbf{ says } delete(file_1)$$

[5] Up to authors knowledge, like all existing formal access control logics do.

Since $\varphi(x)$ and ψ can share variables, we can put restrictions on the variables occurring in ψ. E.g., the set of all users who all own file(s) y asks to delete the file(s) y.

$$\{x\}(user(x) \wedge own(x,y)) \textbf{ says } delete(y)$$

However, the formula above is satisfactory only in the particular situation where we are talking about the set of all users who assert **says** at once as a group (committee).

We can as well express that each member of a set identified by a formula can assert **says** separately. E.g., each user deletes individually the files he owns:

$$\forall x(user(x) \wedge own(x,y)) \rightarrow \{z\}(z = x) \textbf{ says } delete(y)$$

Note that the latter formula usually implies the former but not vice versa[6].

The former formula,

$$\{x\}(\text{user}(x) \wedge \text{ own}(x,y)) \textbf{ says } \text{ delete}(y)$$

expresses the fact that the group of users who own y, i.e. all the owners of y decide (or **say**) as a group to delete y. So maybe they called a meeting, discussed the matter and then had a secret vote. The majority voted to delete y but some voted not to delete. The group outcome was to delete.

The second formula

$$\forall x(\text{user}(x) \wedge \text{ own}(x,y) \rightarrow \{z\}(z = x) \textbf{ says } \text{ delete}(y))$$

expresses the fact that each user who owns y **says** to delete it. This usually implies that the users as a group would **say** to delete y but not necessarily (see footnote 3). Concerning the majority vote example, it may be the case that it is impossible to convene enough users to have a vote and so the set of users never manages to " **say** " as a group to delete y.

Operations on principals We can express the fact that two principals A and B together says **s** :

$$\{x\}(x = A \vee x = B) \textbf{ says s}$$

which corresponds to

$$\{A, B\} \textbf{ says s}$$

[6] In fact, it could be sensible to have situations in which if all the members of a group say something then the whole group says it but not vice versa.

$$\forall t(\varphi(t) \rightarrow t \textbf{ says } \psi) \rightarrow \{x\}\varphi(x) \textbf{ says } \psi$$

For instance, a committee may approve a paper that not all of its members would have accepted.

If we want to express that the intersection of two different kind of principals (T_1,T_2) **says** ψ:

$$\exists x(T_1(x) \wedge T_2(x)) \rightarrow \{y\}(y = x) \textbf{ says } \psi^7$$

with this approach we can also have negation in selecting principals:

$$\{x\}(x \neq A) \textbf{ says s}$$

Variables over principals The possibility to express principals as variables allows first of all attribute-based (as opposed to identity-based) authorization as in [2]. Attribute-based authorization enables collaboration between parties whose identities are initially unknown to each other. The authority to assert that a subject holds an attribute (such as being a student) may then be delegated to other parties, who in turn may be characterised by attributes rather than identity. In the example below, a shop gives a discount to students. The authority over the student attribute is delegated to holders of the university attribute, and authority over the university attribute is delegated to known principal, the Board of Education.

Shop says x is entitled to discount if x is a student.

$$\begin{gathered} \mathit{Shop} \textbf{ says } (\mathit{student}(x) \rightarrow \\ \{y\}(x = y) \textbf{ controls } \mathit{discount}) \end{gathered}$$

Shop says x can say z is a student if x is a university

$$\begin{gathered} \mathit{Shop} \textbf{ says } (\mathit{university}(x) \rightarrow \\ \{y\}(x = y) \textbf{ controls } \mathit{student}(z)) \end{gathered}$$

Shop says BoardOfEducation can say x is a university

$$\mathit{Shop} \textbf{ says } (\mathit{BoardOfEducation} \textbf{ controls } \mathit{university}(x))$$

We may have more complicated policies involving more that two principals, like in the following example [3].

$$\begin{gathered} \{y\}(y = A) \textbf{ says } (((\{y\}(y = C) \textbf{ says } \mathit{fraudulent}(x)) \wedge \\ \{y\}(y = D) \textbf{ says } \mathit{expert}(C)) \rightarrow \mathit{fraudulent}(x)) \end{gathered}$$

Since φ in $\{x\}\varphi(x)$ **says** ψ can be any formula, it can contain even occurrences of the **says** operator. This allows to refer to principals who made previous assertions of the **says** operator. For example, we can express the following: the members of the board who said to write a file they own, ask to delete it.

In symbols

$$\begin{gathered} \{x\}[\{u\}\mathit{member\text{-}board}(u) \textbf{ says } ((\mathit{member\text{-}board}(x) \wedge \\ \mathit{file\text{-}owner}(y, x)) \rightarrow \mathit{write}(y))] \\ \textbf{says } \mathit{delete}(y) \end{gathered}$$

7 For instance, T_1 could be *club_member* and T_2 *adult*.

Like in [2] delegation can be restricted to principals respecting some requirements: Fileserver is a trusted principal who delegates file reading authorizations only to the owners of files:

$$\forall x \; own(x,y) \rightarrow (Fileserver \textbf{ says } (\{z\}(z=x) \\ \textbf{says } read(y) \rightarrow Fileserver \textbf{ says } read(y)))$$

Variables over principals allow width-bounded delegation. Suppose A wants to delegate authority over *is a friend* fact to Bob. She does not care about the length of the delegation chain, but she requires every delegator in the chain to satisfy some property, e.g. to possess an email address. Principals with the *is a delegator* attribute are authorized by A to assert *is a friend* facts, and to transitively re-delegate this attribute, but only amongst principals with a matching email address.

A says x can say y is a friend if x is a delegator

$$A \textbf{ says } ((delegator(x) \rightarrow \\ (\{y\}(x=y) \textbf{ says } friend(z))) \rightarrow friend(z)$$

A says B is a delegator

$$A \textbf{ says } delegator(B)$$

A says x can say y is a delegator if x is a delegator, y possesses email.

$$A \textbf{ says } ((delegator(x) \wedge has\text{-}email(y)) \rightarrow \\ (\{w\}(w=x) \textbf{ controls } delegator(y)))$$

As with depth-bounded delegation, this property cannot be enforced in SPKI/SDSI, DL or XrML.

Restrictions on says Another issue concerns restrictions on speaks for on some issues. Some authors restrict $\Rightarrow$ to a set of propositions [15]

$P \Rightarrow_T Q$ means that the proposition **s** in P **says s** $\rightarrow Q$ **says s** must belong to T.

We can put some restrictions on the variables:

$$(\{x\}(user(x) \wedge owns(x,y)) \textbf{ says } delete(y)) \rightarrow \\ (\{z\}(super\text{-}user(z)) \textbf{ says } delete(y)$$

Moreover we can use the following to restrict the scope of speaks for:

$$\alpha(X) \wedge \Box_B X \rightarrow \Box_A X$$

If $\alpha(X) = \varphi \rightarrow X$ then B speaks for A only on consequences of φ.

The restricted speaks for is strictly related with delegation, if for instance $B \Rightarrow_T A$ we say that B is delegated by A on T. If we want to limit the delegation chain to one step such that we do not permit B to delegate another principal C on T, we add the following constraint:

$$(C \Rightarrow_T B \Rightarrow_T A) \rightarrow (C = B)$$

Separation of duties One of the main concerns in security is the separation of duties: for example the principal(s) signing an order cannot be the same principals who approve it:

$$\neg(\{x\}(\{y\}(x = y) \textbf{ says } \mathit{sign}(\mathit{project})) \textbf{ says } \\ \mathit{approving}(\mathit{project}))$$

In this formula we exploit the full potentiality of FSL in that the principal is defined in terms of the **says** operator.

As noticed in [2] separation of duties requires using negation.

Roles When roles are considered, it emerges the question whether we consider roles types or instances. We distinguish here among roles instances which can be principals by themselves or properties of other principals. So a sentence like "A, who plays a role x of type R, **says s**" becomes:

$$\forall x(x = A \wedge \mathit{role\text{-}played\text{-}by}(x, y) \wedge R(y)) \rightarrow \\ \{z\}(z = y) \textbf{ says s}$$

As concerns hierarchies:

$$\forall x \ \mathit{super\text{-}user}(x) \rightarrow \mathit{user}(x)$$

then

$$\forall x \ \mathit{super\text{-}user}(x) \rightarrow (\{z\}(x = z) \textbf{ says s}) \rightarrow \\ (\forall x \ \mathit{user}(x) \rightarrow (\{z\}(x = z) \textbf{ says s})$$

Instead

$$(\{x\}\mathit{super\text{-}user}(x) \textbf{ says s}) \rightarrow (\{x\}\mathit{user}(x) \textbf{ says s})$$

is less useful: if all super-users say **s** than all users say **s**.

In Abadi [6] if A says something in a role, then it is true that he is playing a role. However, he admits that there should be some requirements to play a role.

For instance, we require that a super-user is a technician:

$$\forall x \ \mathit{super\text{-}user}(x) \rightarrow \mathit{technician}(x)$$

then we can say

$$\forall x \ (x = A \wedge \mathit{super\text{-}user}(x)) \rightarrow (\{z\}(x = z) \textbf{ says s})$$

but there can be no super-user x if A is not a technician.

Parameterized roles can add significant expressiveness to a role-based system and reduce the number of roles [2,13,14]. If we model roles as instances they can have attributes. For instance the example in [2] "NHS[8] says x can access health record of patient if x is a treating clinician of patient" can be modeled as:

[8] National Health Service.

$$\begin{gathered}(\mathit{clinician\text{-}role}(x) \wedge \mathit{patient}(p) \wedge \mathit{record}(r,p) \wedge \\ \mathit{treats}(x,p)) \rightarrow \\ (\{w\}(w = x) \textbf{ says } \mathit{access}(r) \rightarrow \mathit{NHS} \textbf{ says } \mathit{access}(r)))\end{gathered}$$

The operator used to represent a principal A in the role B ($A \mid B$) in [6] is modeled in this way.

$$(A \mid B) \textbf{ says s } \equiv A \textbf{ says } (B \textbf{ says s})$$

In order to match the predicate role-played-by with the above definition we can add the following (where x is a role):

$$\begin{gathered}\forall x, y \ \mathit{role\text{-}played\text{-}by}(x,y) \rightarrow \\ ((x \textbf{ says s}) \rightarrow \\ y \textbf{ says } (\{z\}(z = x) \textbf{ says s}))\end{gathered}$$

Discretionary access control Discretionary access control allows users to pass on their access rights to other users at their own discretion. For instance we can express: "FileServer says user can say x can access resource if user can access resource"[2]

$$\begin{gathered}\forall x \ \mathit{user}(x) \wedge \mathit{user}(z) \rightarrow (\mathit{Fileserver} \textbf{ says} \\ \{w\}(\{y\}(w = y = x) \textbf{ controls } \mathit{access}(u)) \textbf{ controls} \\ \{t\}(t = z) \textbf{ controls } \mathit{access}(u))\end{gathered}$$

Groups In FSL you have to possibility to express how the set $\{x|\varphi(x) \text{ holds}\}$ says what it says, e.g. If $\varphi(x) = (x = A_1) \vee (x = A_2) \vee (x = A_3)$ then if at least one of $\{A_i\}$ **says** ψ is enough for the group to **say** ψ we add:

$$\{x\}\varphi(x) \textbf{ says } \psi \leftrightarrow \bigvee_i \{x\}(x = A_i) \textbf{ says } \psi.$$

This represents the fact that each principal in the group can speak for the whole group. We can as well express that every group has a spokesman (maybe several ones dependent on issues), that one cannot be a spokesman for two different groups and that a group controlling an issue cannot control issues inconsistent with the definition of the group. We can define groups using what they say as part of the definition, put restriction on what they further say or control.

1. *Every group has a spokesman.*
   This is an axiom schema in φ. Let $\textbf{spoke}(\varphi, y)$ be

   $$\begin{gathered}\textbf{spoke}(\varphi, y) = (\forall X[\{x\}\varphi(x) \textbf{ says } X \leftrightarrow \\ \{x\}(x = y) \textbf{ says } X])\end{gathered}$$

   We then take the axiom as $\exists y \ \textbf{spoke} \ (\varphi, y)$.

2. *One cannot be a spokesman for two different groups.*

$$\forall y[\ \textbf{spoke}\ (\varphi_1, y) \wedge\ \textbf{spoke}\ (\varphi_2, y) \rightarrow \\ \forall x[\varphi_1(x) \leftrightarrow \varphi_2(x)]]$$

3. A group cannot control issues inconsistent with the definition of the group

$$\frac{\vdash \varphi \wedge \psi \rightarrow \bot}{\vdash [(\{x\}\varphi(x)\ \textbf{says}\ \psi) \rightarrow \psi] \rightarrow \bot}$$

The following additional axiom expresses that the group identified by the extension of $\{x\}\varphi(x)$ says ψ if at least two members says ψ:

$$\{x\}(\bigvee_i x = A_i)\ \textbf{says}\ \psi \text{ iff} \\ \bigvee_{i \neq j}[\{x\}(x = A_i)\ \textbf{says}\ \psi \wedge \{x\}(x = A_j)\ \textbf{says}\ A_j]$$

More generally, majority voting in $\{x\}\varphi(x)$ **says** ψ, is just an axiom.

$$\{x\}\varphi(x)\ \textbf{says}\ \psi \leftrightarrow \bigvee_i \{x\}\varphi_i(x)\ \textbf{says}\ \psi$$

where $\varphi_i(x)$ are all formulas $(\forall x \varphi_i(x) \rightarrow \varphi(x))$ defining majorities in the set $\{x\}\varphi(x)$.

Majority vote is an example of threshold-constrained trust SPKI/SDSI [12]. The concept of k-of-n threshold subjects means that at least k out of n given principals must sign a request and it is used to provide a fault tolerance mechanism. RTT has the language construct of "threshold structures" for similar purposes [39]. As in SecPAL [2] there is no need for a dedicated threshold construct, because threshold constraints can be expressed directly.

4 The basic system FSL

This section introduces our basic system FSL step by step from a semantics point of view. First we introduce modalities indexed by propositional atoms, then we take into account classical and intuitionistic models for the propositional setting and finally we give semantics to predicate FSL that we extensively employed in previous sections.

This system can be defined with any logic $\mathbb{L}$ as a *Fibred Security System based on* $\mathbb{L}$. We will motivate the language for the cases of $\mathbb{L}$ = classical logic and $\mathbb{L}$ = intuitionistic logic.

Basically adding the **says** connective to a system is like adding many modalities. So to explain and motivate FSL technically we need to begin with examining options for adding modalities to $\mathbb{L}$. Subsection 4.1 examines our options of how to add modalities to classical and intuitionistic logics. The presentation and discussion is geared towards subsection 4.2 which presents FSL.

4.1 Adding modalities

We start by adding modalities to classical propositional logic. We are going to do it in a special way. The reader is invited to closely watch us step-by-step,

Our approach is semantic.

Let S be a nonempty set of possible worlds. For every subset $U \subseteq S$ consider a binary relation $R_U \subseteq S \times S$.

This defines a multi-modal logic, containing K modalities $\Box_U, U \subseteq S$. The models are of the form $(S, R_U, t_0, h), U \subseteq S$. In this view, if $U = \{t | t \vDash \varphi_U\}$ for some φ_U we get a modal logic with modalities indexed by formulas of itself. This requires now a formal definition.

Definition 2 (Language). *Consider (classical or intuitionistic) propositional logic with the connectives $\wedge, \vee, \rightarrow, \neg$ and a binary connective $\Box_\varphi \psi$, where φ and ψ are formulas.*[9] *The usual definition of wff is adopted.*

Definition 3. *We define classical Kripke models for this language.*

1. *A model has the form*

$$\mathbf{m} = (S, R_U, t_0, h), U \subseteq S$$

where for each $U \subseteq S, R_U$ is a binary relation on S. $t_0 \in S$ is the actual world and h is an assignment, giving for each atomic q a subset $h(q) \subseteq S$.

2. *We can extend h to all formulas by structural induction:*
   - *$h(q)$ is already defined, for q atomic*
   - *$h(A \wedge B) = h(A) \cap h(B)$*
   - *$h(\neg A) = S - h(A)$*
   - *$h(A \rightarrow B) = (S - h(A)) \cup h(B)$*
   - *$h(A \vee B) = h(A) \cup h(B)$*
   - *$h(\Box_\varphi \psi) = \{t |$ for all s $(t R_{h(\varphi)} s \rightarrow s \in h(\psi))\}$*
3. *$\mathbf{m} \vDash A$ iff $t_0 \in h(A)$.*

There is nothing particularly new about this except possibly the way we are looking at it.

Let us now do the same for intuitionistic logic. Here it becomes more interesting. An intuitionistic Kripke model has the form

$$\mathbf{m} = (S, \leq, t_0, h),$$

where $(S, \leq)$ is a partially ordered set, $t_0 \in S$ and h is an assignment to the atoms such that $h(q) \subseteq S$. We require that $h(q)$ is a closed set, namely

- $x \in h(q)$ and $x \leq y$ imply $y \in h(q)$.

[9] There are many such connectives, e.g. φ **says** $\psi, \varphi > \psi$ (conditional), $\bigcirc(\varphi/\psi)$ relative obligation, etc. The semantics given to it will determine its nature.

Let D be a set and we can add for each $U \subseteq D$ a binary relation R_U on S. This semantically defines an intuitionistic modality, $\Box_U$.

In intuitionistic models we require the following condition to hold for each formula A, i.e. we want $h(A)$ to be closed:

– $x \in h(A)$ and $x \leq y \Rightarrow y \in h(A)$

This condition holds for A atomic and propagates over the intuitionistic connectives $\wedge, \vee, \rightarrow, \neg, \bot$. To ensure that it propagates over $\Box_U$ as well, we need an additional condition on R_U. To see what this condition is supposed to be, assume $t \vDash \Box_U A$. This means that

$$\forall y(tR_U y \Rightarrow y \vDash A).$$

Let $t \leq s$. If $s \nvDash \Box_U A$, then for some z such that $sR_U z$ we have $z \nvDash A$. This situation will be impossible if we require

$$t \leq s \wedge sR_U z \Rightarrow tR_U z. \qquad (*)$$

Put differently, if we use the notation:

$$R'_U(x) = \{y | xR_U y\}$$

then

$$x \leq x' \Rightarrow R'_U(x) \supset R'_U(x'). \qquad (*)$$

So we now talk about modalities R_U, for $U \subseteq S$. We ask what happens if U is defined by a formula φ_U, i.e. $U = h(\varphi_U)$. This will work only if U is closed

– $t \in U \wedge t \leq s \Rightarrow s \in U$.

So from now on, we talk about modalities associated with closed subsets of S.

We can now define our language. This is the same as defined in Definition 2. We now define the semantics.

Definition 4. *A model has the form*

$$\mathbf{m} = (S, \leq, R_U, t_0, h), U \subseteq S$$

where $(S, \leq)$ is a partial order, $t_0 \in S$, and each $U \subseteq S$ is a closed set and so is $h(q)$ for atomic q. R_U satisfies condition (*) above. We define the notion $t \vDash A$ for a wff by induction, and then define

$$h(A) = \{t | t \vDash A\}.$$

So let's define $\vDash$:

– $t \vDash q$ *iff* $t \in h(q)$
– $t \vDash A \wedge B$ *iff* $t \vDash A$ *and* $t \vDash B$
– $t \vDash A \vee B$ *iff* $t \vDash A$ *or* $t \vDash B$
– $t \vDash A \rightarrow B$ *iff for all* $s, t \leq s$ *and* $s \vDash A$ *imply* $s \vDash B$

– $t \models \neg A$ *iff for all* s, $t \leq s$ *implies* $s \not\models A$
– $t \not\models \bot$
– $t \models \Box_\varphi \psi$ *iff for all* s *such that* $tR_{h(\varphi)}s$ *we have* $s \models \psi$. *We assume by induction that* $h(\varphi)$ *is known.*
– $\mathbf{m} \models A$ *iff* $t_0 \models A$.

It is our intention to read $\Box_\varphi \psi$ as φ **says** ψ.

Example 1 (Two intuitionistic modalities). Let us examine the case of two intuitionistic modalities in more detail Let us call them $\Box_A$ and $\Box_B$ and their accessibility relations R_A and R_B. So our Kripke model has the form $(S, \leq, R_A, R_B, t_0, h)$. We know for $\mu = A$ or $\mu = B$ that we have in the model

$$t \leq s \wedge sR_\mu z \to tR_\mu z. \tag{$*$}$$

What other conditions can we impose on $\Box_\mu$?

1. *The axiom* $X \to \Box_\mu X$
   This corresponds to the condition
   $$xR_\mu y \to x \leq y \tag{$*1$}$$
2. *The axiom* $\Box_B X \to \Box_A X$
   This corresponds to the condition
   $$xR_A y \to xR_B y \tag{$*2$}$$
3. Note that $\Box_B X \to \Box_A X$ is taken in (*2) as an axiom schema. If we want to have $t \models \forall X(\Box_B X \to \Box_A X)$ i.e. we want $\Box_B \varphi \to \Box_A \varphi$ to hold at the point $t \in S$ for all wff φ, we need to require (*2) to hold above t, i.e.
   $$\forall x, y(t \leq x \wedge xR_A y \to xR_B y) \tag{$*3$}_t$$
4. Consider now an axiom called *hand-off A to B*.
   $$\Box_A(\forall X(\Box_B X \to \Box_A X)) \to \forall X(\Box_B X \to \Box_A X)$$
   This axiom has a second order propositional quantifier in it.
   The antecedent of the axiom wants $\Box_A(\forall X \Box_B X \to \Box_A X))$ to hold at t_0. This means in view of (3) above that $(*4_a)$ needs to hold
   $$\forall t(t_0 R_A t \to (*3)_t) \tag{$*4_a$}$$
   The axiom says that if the antecedent holds at t_0 so does the consequent, i.e.
   $$t_0 \models \forall X(\Box_B X \to \Box_A X).$$
   We know the condition for that to hold is $(*3)_{t_0}$. Thus the condition for Hand-off A to B is
   $$\forall t[t_0 R_A t \to (*3)_t] \to (*3)_{t_0} \tag{$*4$}$$

The important point to note is that although the axiom is second order (has $\forall X$ in it both in the antecedent and consequence), the condition on the model is first order[10].

5. There is another modal axiom called escalation for A

$$\Box_A X \to X \vee \Box_A \bot$$

The condition for that is

$$\exists y(xR_A y) \to xR_A x \tag{$*5$}$$

To check whether we can have hand-off from A to B without escalation for A, for some choice of R_A and R_B, we need to check whether we can have (*4) without having (*5), for some wise choice of R_A and R_B.

6. Consider a Kripke model $(S, \leq, t_0)$ which is nonending and dense, i.e.
   – $\forall x \exists y(x \lneqq y)$
   – $\forall xy(x \lneqq y \to \exists z(x \lneqq z \lneqq y))$

   In this model let

$$xR_A y \text{ be } x \lneqq y$$
$$xR_B y \text{ be } x \leq y.$$

We have here that $(*3)_t$ holds for any t because it says

$$\forall xy(t \leq x \wedge x \lneqq y \to x \leq y)$$

Therefore (*4) also holds. This is hand-off from A to B.
However, escalation does not hold because

$$\exists y(x \lneqq y) \to x \lneqq x$$

is false.

7. *α relative hand-off*
   Let $\alpha(x)$ be any formula and X a new variable. We can write a relative speak axiom.

$$\alpha(X) \wedge \Box_B X \to \Box_A X \tag{$*7$}$$

In particular, if $\alpha(x) = \varphi \to x$, then (*7) would refer to B speaks for A on all consequences of φ.

Definition 5. *Let $(S, \leq, t_0, h)$ be a Kripke model. By a modal function* $\mathbf{E}$ *we mean a function giving to each point $t \in S$ a set of points $\mathbf{E}(t)$ such that*

1. *$t \lneqq s$ for all $s \in \mathbf{E}(t)$.*
2. *$t_1 \leq t_2 \to \mathbf{E}(t_1) \leq \mathbf{E}(t_2)$ where $\mathbf{E}(t_1) \leq \mathbf{E}(t_2)$ means $\forall x \in \mathbf{E}(t_2) \exists y \in \mathbf{E}(t_1)(y \leq x)$.*

Definition 6. $(S, \leq, t_0, \mathbf{E})$ *is* $\mathbf{E}$*-dense iff the following holds:*

[10] Notice that we use first-order but we get a language more expressive than CDD[9] which is second-order.

– *If $x \in \mathbf{E}(t)$, then for some $y, y \in \mathbf{E}(t) \wedge x \in \mathbf{E}(y)$.*

To show the existence of dense systems, we do the following construction:

1. *Start with $(S_0, \leq_0, \mathbf{E}_0)$ where $\mathbf{E}_0(x)$ is dense. For example, $(S_0, \leq_0)$ may be linear and $\mathbf{E}_0$ is generated by a strictly increasing function $\mathbf{f}$, i.e. $\mathbf{E}(x) = \{y | \mathbf{f}(x) \leq y\}$.*
2. *For every pair of points $x \lneqq_0 y$, add the point (x, y).*
   Let $x \lneqq_0 (x,y) \lneqq_0 y$ and let $S_1 = S_0 \cup \{(x,y) | x \lneqq_0 y\}$, let $\leq_1$ be transitive closure of $\leq_0 \cup \{(x,(x,y)), ((x,y),y)\}$. Let $\mathbf{E}_1$ be defined from $\leq_1$ as follows: First let $\bar{\mathbf{E}}$ denote the $\leq_1$ closure of $\mathbf{E}$.

   $$x \in \bar{\mathbf{E}} \textit{ iff } \exists y \in \mathbf{E} \textit{ such that } y \leq_1 x.$$

   Second, let for each x

   $$P(x) = \{(x,y) | x \leq_0 y \textit{ and } x \neq y\}$$

   Then let

   $$\mathbf{E}_1(x) = \overline{\mathbf{E}_0(x) \cup P(x)}$$
   $$\mathbf{E}_1((x,y)) = \{z | y \leq_1 z\}.$$
3. *Let $(S_{n+1}, \leq_{n+1}, t_0, \mathbf{E}_{n+1})$ be obtained from $S_{n+1}, \leq_{n+1}$ in the same way as in step 2.*
4. *Let $(S_\infty, \leq_\infty, t_0, \mathbf{E}_\infty)$ be the union.*

   $$S_\infty = \bigcup_n S_n, \leq_\infty = \bigcup_n \leq_n, \mathbf{E}_\infty = \bigcup_n \mathbf{E}_n.$$

   Then we have density.
   If $y \in \mathbf{E}_\infty(x)$ then for some z

   $$z \in \mathbf{E}_\infty(x) \wedge y \in \mathbf{E}_\infty(z)$$

Remark 1. In models of Definition 6 we have *Unit* and *C4* hold but not necessarily *Escalation* nor *Generalised Hand-off.*

4.2 Predicate FSL

Intuitively, a predicate FSL fibred model is represented by a set of models linked togheter by means of a *fibring function*, every model has an associated domain D of elements together with a set of formulas that are true in it. In the FSL meta-model, the evaluation of the generic formula $\{x\}\varphi(x)$ **says** is carried out in two steps, first evaluating φ and then ψ in two different models. Suppose $\mathbf{m_1}$ is our (first order) starting model in which we identify $U \subseteq D$ as the set of all the elements that satisfy φ. Once we have U we can access one or more worlds depending on the *fibring function* $\mathbf{f} : \mathcal{P}(D) \rightarrow \mathcal{P}(M)$ which goes from sets of elements in domain D to sets of models. A this point, for every model $\mathbf{m_i} \in \mathbf{f}(U)$ we must check that ψ is *true*, if this is the case then α is true in the meta-model.

The fact that in the same expression we evaluate different sub-formulas in different models it is not completely counter intuitive, for instance, think about a group of administrators that have to set up security policies for their company. From a semantical point of view, if we want to check if ψ holds in the depicted configuration by the administrators, we must

1. Identify all the admins (all the elements that satisfy $admin(x)$).
2. Access the model that all the admins as a group have depicted.
3. Check in that model if ψ is *true* or *false*

Let $\mathbb{L}$ denote classical or intuitionistic predicate logic.[11] We assume the usual notions of variables, predicates, connectives $\wedge, \vee, \rightarrow, \neg$, quantifiers $\forall, \exists$ and the notions of free and bound variables.

Let $\mathbb{L}^+$ be $\mathbb{L}$ together with two special symbols:

– A binary (modality), x **says** y
– A set-binding operator $\{x\}\varphi(x)$ meaning the set of all x such that $\varphi(x)$

Note that semantically at the appropriate context $\{x\}\varphi(x)$ can behave like $\forall x\varphi(x)$ and sometimes in other contexts, we will use it as a set.

Definition 7. *The language FSL has the following expressions:*

1. *All formulas of $\mathbb{L}^+$ are level 0 formulas of FSL.*
2. *If $\varphi(x)$ and ψ are formulas of $\mathbb{L}^+$ then $\alpha = \{x\}\varphi(x)$ **says** ψ are level 1 'atomic' formulas of FSL. If $(x, x_1, \ldots, x_n)$ are free in φ and $y_1, \ldots, y_m$ are free in ψ then $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ are free in α. The variable x in φ gets bound by $\{x\}$. The formula of level 1 are obtained by closure under the connectives and quantifiers of $\mathbb{L}^+$.*
3. *Let $\varphi(x)$ and ψ be formulas of FSL of levels r_1 and r_2 resp., then $\alpha = \{x\}\varphi$ **says** ψ is an 'atomic' formula of FSL of level $r = \max(r_1, r_2) + 1$.*
4. *Formulas of level n are closed under classical logic connectives and quantifiers of all 'atoms' of level $m \leq n$.*

Definition 8 (FSL classical fibred model of level n).

1. *Any classical model with domain D is an FSL model of level 0.*
2. *Let $\mathbf{m}$ be a classical model of level 0 with domain D and let for each subset $U \subseteq D, \mathbf{f}^n(U)$ be a family of models of level n (with domain D). Then $(\mathbf{m}, \mathbf{f}^n)$ is a model of level $n+1$.*

Definition 9 (Classical satisfaction for FSL). *We define satisfaction of formulas of level n in classical models of level $n' \geq n$ as follows.*

*First observe that any formula of level n is built up from atomic predicates of level 0 as well as 'atomic' formulas of the form $\alpha = \{x\}\varphi(x)$ **says** ψ, where φ and ψ are of lower level.*

[11] Classical predicate logic and intuitionistic predicate logic have the same language. The difference is in the proof theory and in the semantics.

We therefore first have to say how we evaluate $(\mathbf{m}, \mathbf{f}^n) \vDash \alpha$.

We assume by induction that we know how to check satisfaction in $\mathbf{m}$ *of any* $\varphi(x)$, *which is of level* $\leq n$.

We can therefore identify the set $U = \{d \in D \mid \mathbf{m} \vDash \varphi(d)\}$.

Let $\mathbf{m}' \in \mathbf{f}^n(U)$. *We can now evaluate* $\mathbf{m}' \vDash \psi$, *since* ψ *is of level* $\leq n-1$.

So we say

$$(\mathbf{m}, \mathbf{f}^n) \vDash \alpha \textit{ iff for all } \mathbf{m}' \in \mathbf{f}^n(U), \textit{ we have } \mathbf{m}' \vDash \psi.$$

We need to add that if we encounter the need to evaluate $\mathbf{m} \vDash \{x\}\beta(x)$, *then we regard* $\{x\}\beta(x)$ *as* $\forall x \beta(x)$.

Example 2. Figure 1 is a model for

$$\alpha(y) = \{x\}[\{u\}B(u) \textbf{ says } (B(x) \rightarrow A(x,y))] \textbf{ says } F(y)$$

In Figure 1, $\mathbf{m}_1$ is a single model in $\mathbf{f}^1(U_B)$ and $\mathbf{m}_3$ is a single model in $\mathbf{f}^1(U_{E(y)})$, as defined later.

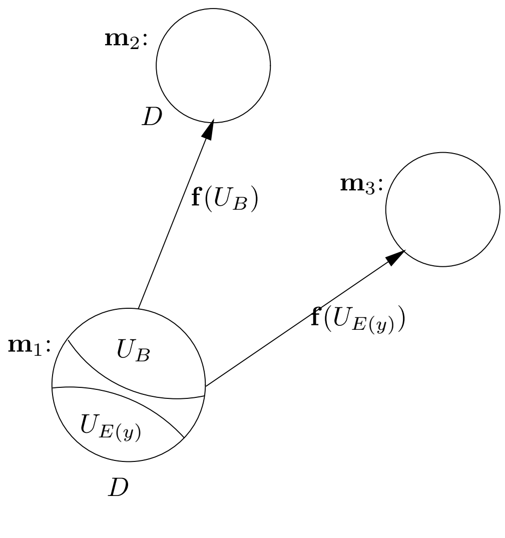

Fig. 1.

The set U_B is the extension of $\{x\}B(x)$ in $\mathbf{m}_1$.

To calculate the set of pairs (x, y) such that $E(x, y) = \{u\}B(u)$ **says** $(B(x) \rightarrow A(x, y))$ holds in $\mathbf{m}_1$, we need to go to $\mathbf{m}_2$ in $\mathbf{f}(U_B)$ and check whether $B(x) \rightarrow A(x, y)$ holds in $\mathbf{m}_2$, x, y are free variables so we check the value under fixed assignment.)

We now look at $E(y) = \{x\}E(x, y)$ for y fixed, we collect all elements d in D such that $\mathbf{m}_2 \vDash B(d) \rightarrow A(d, y)$. Call this set $U_{E(y)}$.

To check $\alpha(y) = \{x\}E(x, y)$ **says** $F(y)$ in $\mathbf{m}_1$ we have to check whether $F(y)$ holds in $\mathbf{m}_3$.

We now define intuitionistic models for FSL. This will give semantics for the intuitionistic language.

Definition 10. *We start with intuitionistic Kripke models which we assume for simplicity have a constant domain. The model $\mathbf{m}$ has the form $(S, \leq, t_0, h, D)$ where D is the domain and $(S, \leq, t_0)$ is a partial order with first point t_0 and h is an assignment function giving for each $t \in S$ and each m-place atomic predicate P a subset $h(t, P) \subseteq D^m$ such that $t_1 \leq t_2 \Rightarrow h(t_1, P) \subseteq h(t_2, P)$*

We let $h(P)$ denote the function $\lambda t\ h(t, P)$. For $t \in S$ let

$$\begin{array}{l} S_t = \{s \mid t \leq s\} \\ h(t, P) = h(P) \restriction S_t \\ \leq_t = \leq \restriction S_t \end{array}$$

Let $\mathbf{m}_t = (S_t, \leq_t, t, h_t, D)$.

Note that a formula φ holds at $\mathbf{m} = (S, \leq, t_0, h, D)$ iff $t_0 \vDash \varphi$ according to the usual Kripke model definition of satisfaction.

1. *A model of level 0 is any model $\mathbf{m}$: $\mathbf{m} = (S, \leq, t_0, h, D)$.*
2. *Suppose we have defined the notion of models of level $m \leq n$, (based on the domain D).*

We now define the notion of a model of level $n + 1$

Let $\mathbf{m}$ be a model of level 0 with domain D. We need to consider not only $\mathbf{m}$ but also all the models $\mathbf{m}_t = (S_t, \leq_t, t, h_t, D)$, for $t \in S$. The definitions will be given simultaneously for all of them.

By an intuitionistic 'subset' of D in $(S, \leq, t_0, h, D)$, we mean a function $\mathbf{d}$ giving for each $t \in S$, a subset $\mathbf{d}(t) \subseteq D$ such that $t_1 \leq t_2 \Rightarrow \mathbf{d}(t_1) \subseteq \mathbf{d}(t_2)$.

Let $\mathbf{f}^n_t$ be a function associating with each $\mathbf{d}_t$ and $t \in S$ a family $\mathbf{f}^n_t(\mathbf{d}_t)$ of level n models, such that $t_1 \leq t_2 \Rightarrow \mathbf{f}^n_{t_1}(\mathbf{d}_{t_1}) \supseteq \mathbf{f}^n_{t_2}(\mathbf{d}_{t_2})$. Then $(\mathbf{m}_t, \mathbf{f}_t)$ is a model of level $n + 1$ where $\mathbf{d}_t = \mathbf{d} \restriction S_t$.

Definition 11 (Satisfaction in fibred intuitionistic models). *We define satisfaction of formulas of level n in models of level $n' \geq n$ as follows.*

*Let $(\mathbf{m}_t, \mathbf{f}^n_t)$ be a level n model. Let $\alpha = \{x\}\varphi(x)$ **says** ψ is of level n. We assume we know how to check satisfaction of $\varphi(x)$ in any of these models.*

We can assume that

$$\mathbf{d}_t = \{x \in D \mid t \vDash \varphi(x) \text{ in } (\mathbf{m}_t, \mathbf{f}^n_t)\}$$

is defined. Then $t \vDash \alpha$ iff for all models $\mathbf{m}'_t$ in $\mathbf{f}^n_t(\mathbf{d}_t)$ we have $\mathbf{m}'_t \vDash \psi$.

5 An Example

In this section we want to give an informal example of policy written in FSL. Suppose we have the following distributed policies and facts of a computer science department:

- The department of Computer Science (CS_Dep) delegates the $University$ to say who is regularly enrolled as a student

$$\begin{gathered}\forall x(University \textbf{ says } regularly\_enrolled(x) \rightarrow_{12} \\ CS\_Dep \textbf{ says } regularly\_enrolled(x))\end{gathered} \tag{3}$$

- The delegation depth between $University$ and CS_Dep must be limited to one, we have that for every principal P ($P \in D$):

$$\begin{gathered}P \Rightarrow_{regularly\_enrolled(x)} University \rightarrow \\ P = University\end{gathered} \tag{4}$$

- The CS_Dep delegates the *group* of all the members of the ICT staff to assign logins to students

$$\begin{gathered}\forall y(\{x\}ICT\_member(x) \textbf{ says } has\_login(y) \rightarrow \\ CS\_Dep \textbf{ says } has\_login(y))\end{gathered} \tag{5}$$

- The group of the ICT staff members says that z has a login if and only if one single member of the group says it

$$\begin{gathered}\{x\}ICT\_member(x) \textbf{ says } has\_login(z) \leftrightarrow \\ \exists y(ICT\_member(y) \wedge y \textbf{ says } has\_login(z))\end{gathered} \tag{6}$$

- If someone has a login and is regulary enrolled as student at the University then he can have access to his mail:

$$\begin{gathered}(CS\_Dep \textbf{ says } has\_login(x) \wedge \\ CS\_Dep \textbf{ says } regulary\_enrolled(x) \wedge \\ x \textbf{ says } can\_access\_mail(x)) \rightarrow \\ can\_access\_mail(x)\end{gathered} \tag{7}$$

- John and Adam are members of the ICT staff of che Computer Science Departement:

$$ICT\_member(John) \wedge ICT\_member(Adam) \tag{8}$$

- The University certifies that Tom is regulary enrolled:

$$University \textbf{ says } regularly\_enrolled(Tom) \tag{9}$$

[12] CS_Dep **says** ψ in formal FSL must be inteded as $\{x\}(x = CS\_Dep)$ **says** ψ

– Tom has a login which has been assigned by Adam

$$Adam \textbf{ says } has\_login(Tom) \tag{10}$$

Suppose now we want to check if Tom can access his mail, so that from

$$Tom \textbf{ says } can\_access\_mail(Tom) \tag{11}$$

we want, on the basis of the following knowledge base, to derive

$$can\_access\_mail(Tom) \tag{12}$$

In fact we can make the following reasoning:

– From (1) and (6) we get

$$CS\_Dep \textbf{ says } regularly\_enrolled(Tom) \tag{13}$$

– From (7) and (3) we derive

$$\{x\}ICT\_member(x) \textbf{ says } has\_login(Tom) \tag{14}$$

– With (2) and (11) we obtain

$$CS\_Dep \textbf{ says } has\_login(Tom) \tag{15}$$

– Now with (12),(10),(8) and (4) we finally conclude

$$can\_access\_mail(Tom) \tag{16}$$

6 Conclusion and Future Works

In this paper we presented FSL, a language for access control in distributed systems. Our approach is based on fibring [1] which is a methodology to compose logics and use them within a same language. In Section 4.2 we introduced a fibred semantics to merge intuitionistic logic with modalities indexed by first-order formulas creating predicate FSL. Predicate FSL is a language which satisfies the requirements listed in Section 1, in future works we plan to first extend well known existing logics like Delegation Logic [3], SecPAL [2] and DEBAC [4] with the FSL methodology and then to translate them into predicate FSL modal logic. We are also working on providing a calculus for predicate FSL, in order to maintain the calculus tractable we plan to employ first-order modal theorem provers without resorting to second order.

References

1. D. M. Gabbay, "Fibring logics," *Oxford University Press*, 1999.
2. M. Y. Becker, C. Fournet, and A. D. Gordon, "Design and semantics of a decentralized authorization language," in *CSF*. IEEE Computer Society, 2007, pp. 3–15.
3. N. Li, B. N. Grosof, and J. Feigenbaum, "Delegation logic: A logic-based approach to distributed authorization," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 1, pp. 128–171, 2003.
4. C. Bertolissi, M. Fernández, and S. Barker, "Dynamic event-based access control as term rewriting," in *DBSec*, ser. Lecture Notes in Computer Science, S. Barker and G.-J. Ahn, Eds., vol. 4602. Springer, 2007, pp. 195–210.
5. B. W. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems: Theory and practice," *ACM Trans. Comput. Syst.*, vol. 10, no. 4, pp. 265–310, 1992.
6. M. Abadi, M. Burrows, B. W. Lampson, and G. D. Plotkin, "A calculus for access control in distributed systems," in *CRYPTO*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Springer, 1991, pp. 1–23.
7. M. Abadi, "Logic in access control," in *LICS*. IEEE Computer Society, 2003, pp. 228–.
8. ——, "Access control in a core calculus of dependency," *Electr. Notes Theor. Comput. Sci.*, vol. 172, pp. 5–31, 2007.
9. ——, "Variations in access control logic," in *DEON*, ser. Lecture Notes in Computer Science, R. van der Meyden and L. van der Torre, Eds., vol. 5076. Springer, 2008, pp. 96–109.
10. B. W. Lampson, "Computer security in the real world," *IEEE Computer*, vol. 37, no. 6, pp. 37–46, 2004.
11. M. D. Schroeder and J. H. Saltzer, "The protection of information in computer systems," *Procs. IEEE 63*, vol. 9, pp. 1278–1308, 1975.
12. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "Spki certificate theory," *IETF RFC 2693*, 2009.
13. L. Giuri and P. Iglio, "Role templates for content-based access control," in *ACM Workshop on Role-Based Access Control*, 1997, pp. 153–159.
14. E. Lupu and M. Sloman, "Reconciling role based management and role based access control," in *ACM Workshop on Role-Based Access Control*, 1997, pp. 135–141.
15. T. Kosiyatrakul, S. Older, and S.-K. Chin, "A modal logic for role-based access control," in *MMM-ACNS*, ser. Lecture Notes in Computer Science, V. Gorodetsky, I. V. Kotenko, and V. A. Skormin, Eds., vol. 3685. Springer, 2005, pp. 179–193.

A Appendix

A.1 Axiomatisation and completeness of FSL

We prove completeness for FSL with increasing domains and for FSL with constant domains (FSL and FSL_{CD}). Well-formed formulas (wffs) are defined recursively as follows:

- Atoms of the form $P(t_1 \dots t_n)$[13] are wffs.

[13] where $t_1 \dots t_n$ are classical first-order terms.

– $\bot$ is a wff.
– If α and β are wff, then so are $(\neg\alpha), (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \to \beta), (\forall x\alpha), (\exists x\alpha)$.
– If $\varphi(x)$ and ψ are wff, the so is $\{x\}\varphi(x)\,\mathbf{says}\,\psi$.

Axiom system for predicate FSL

1. All axioms and rules for intuitionistic logic
2. Extensionality axiom:

$$\forall x(\varphi_1(x) \leftrightarrow \varphi_2(x)) \to \\ (\{x\}\varphi_1(x)\ \mathbf{says}\ \psi \leftrightarrow \{x\}\varphi_2(x)\ \mathbf{says}\ \psi)$$

3. Modality axioms:

$$\frac{\vdash \bigwedge_i \alpha \to \beta}{\vdash \bigwedge_i \{x\}\varphi\,\mathbf{says}\,\alpha_i \to \{x\}\varphi\,\mathbf{says}\,\beta}$$

4. Constant domains axioms[14]:
   (a) $\forall y\{x\}\varphi\,\mathbf{says}\,\beta(y) \to \{x\}\varphi\,\mathbf{says}\,\forall y\beta(y)$
   (b) $\forall y(\psi \vee \beta(y)) \to (\psi \to \forall y\beta(y))$
5. Additional Axioms:
   (a) $A \to \{x\}\varphi\,\mathbf{says}\,A$
   (b) *here we put all the axioms we need to craft our logic from Sections 2,3 like for instance:*

$$\forall t(\varphi(t) \to t\,\mathbf{says}\,\psi) \to \{x\}\varphi(x)\,\mathbf{says}\,\psi$$

Definitions and Lemmas

Definition 12 (Consistent and Complete Theory). *Suppose we have a theory (Δ, Θ) of sentences*[15]*.*

– *(Δ, Θ) is consistent, if we <u>do not</u> have for some $\alpha_i \in \Delta$, $\beta_j \in \Theta$*

$$\vdash \bigwedge_i \alpha_i \to \bigvee_j \beta_j$$

– *(Δ, Θ) is complete in the language with variables $\mathcal{V}$ iff for all ψ in the language, we have*

$$\psi \in \Delta \text{ or } \psi \in \Theta$$

Definition 13 (Saturated Theory). *A theory (Δ, Θ) is saturated in a language with variables $\mathcal{V}$ iff the following holds:*

1. (Δ, Θ) is consistent

[14] y not free in ψ or φ.
[15] intuitively, Δ is the set of formulas that are true in the model and Θ is the set of formulas that are false in the model.

2. $\exists x A(x) \in \Delta$*, then for some* $y \in \mathcal{V}$*,* $A(y) \in \Delta$*.*
3. $\forall x A(x) \notin \Delta$*, then for some* $y \in \mathcal{V}$*,* $A(y) \notin \Delta$
4. $A \vee B \in \Delta$ *iff* $A \in \Delta$ *or* $B \in \Delta$*.*
5. If for some $\beta_j \in \Theta$

$$\Delta \vdash A \vee \beta_j \Rightarrow A \in \Delta$$

with A *in the language with variables* $\mathcal{V}$

Definition 14 (Constant Domain Theory). *A theory* (Δ, Θ) *is said to be constant domain (CD) theory in language* $\mathcal{V}$ *iff for any* $\forall x A(x)$ *and any* $\beta_j \in \Theta$ *such that*

$$\Delta \nvdash \forall x A(x) \vee \bigvee_j \beta_j$$

then for some y

$$\Delta \nvdash A(y) \vee \bigvee_j \beta_j$$

Lemma 1. *Assume the CD axiom* $\forall x(\beta \vee A(x) \to (\beta \vee \forall x A(x)))$*, then if* (Δ, Θ) *is a consistent CD theory and* $\Delta^{'} = \Delta \cup \{\alpha_1, \ldots, \alpha_n\}$*,* $\Theta^{'} = \Theta \cup \{\gamma_1, \ldots, \gamma_m\}$ *and* $(\Delta^{'}, \Theta^{'})$ *is consistent then* $(\Delta^{'}, \Theta^{'})$ *is a CD theory*

Proof. *Assume*

$$\Delta \cup \bigwedge_i \alpha_i \nvdash (\bigvee_j \beta_j) \vee (\bigvee_j \gamma_j) \vee \forall x A(x)$$

we can assume x *not in* β_j*,*α_j*,* γ_j *hence*

$$\Delta \nvdash \textstyle\bigwedge_i \alpha_i \to \forall x(\bigvee_j \beta_j) \vee (\bigvee_j \gamma_j) \vee \forall x A(x)$$
$$\Delta \nvdash \forall x(\textstyle\bigwedge_i \alpha_i \to (\bigvee_j \beta_j) \vee (\bigvee_j \gamma_j) \vee A(x))$$

hence for some y

$$\Delta \nvdash \bigwedge_i \alpha_i \to \beta \vee A(y) \vee \gamma_j$$

hence $\Delta^{'} \nvdash \beta \vee A(y) \vee \gamma_j$

Lemma 2. *Let* (Δ, Θ) *be a saturated theory. Let* $\Delta^{'}$ *be*

$$\{\psi | (\{x\}\varphi(x) \textbf{ says } \psi) \in \Delta\}$$

Assume

$$(\{x\}\varphi(x) \ \textbf{\textit{says}} \ \beta) \in \Theta$$
$$\Delta^{'} \nvdash \beta \vee \forall x A(x)$$

then for some y

$$\Theta \nvdash \beta \vee A(y)$$

Proof. The proof is by contradiction, suppose it is not the case that

$$\Theta \nvdash \beta \vee A(y)$$

then, for each y there exists a finite $\Delta'_y \subseteq \Delta'$ such that

$$\nvdash \bigwedge \Delta'_y \rightarrow \beta \vee A(y)$$

hence, with $\alpha \in \Delta'_y$

$$\nvdash \bigwedge\{x\}\varphi(x) \textbf{ says } \alpha \rightarrow \{x\}\varphi \textbf{ says } \beta \vee A(y)$$

hence, for all y

$$\{x\}\varphi \textbf{ says } \beta \vee A(y) \in \Delta$$

Since Δ is saturated we get:

$$\forall y\{x\}\varphi \textbf{ says } (\beta \vee A(y)) \in \Delta$$

hence

$$\{x\}\varphi \textbf{ says } \forall y(\beta \vee A(y)) \in \Delta$$

hence

$$\forall y(\beta \vee A(y)) \in \Delta'$$

but then

$$\beta \vee \forall y A(y) \in \Delta'$$

which is a contradiction.

Lemma 3. *Let (Δ, Θ) be a consistent CD theory, then (Δ, Θ) can be extended to a saturated theory (Δ', Θ') in the same laguage with $\Delta \subseteq \Delta'$ and $\Theta \subseteq \Theta'$*

Proof. The proof is by induction on (Δ_n, Θ_n) the theory, let $\Delta_o = \Delta$ and $\Theta_0 = \Theta$.

Assume (Δ_n, Θ_n) is defined, $\Theta_n - \Theta$ is finite and (Δ_n, Θ_n) is CD. Let β_{n+1} be the $(n+1)$th wff of the language. Then either $(\Delta_n, \Theta_n \cup \beta_{n+1})$ is consistent or is not consistent, it it is consistent let

$$\begin{array}{l} \Delta_{n+1} = \Delta_n \\ \Theta_n + 1 = \Theta_n \cup \{\beta\} \end{array}$$

If it is inconsistent then $(\Delta_n \cup \beta, \Theta_n)$ must be consistent so let

$$\begin{array}{l} \Delta_{n+1} = \Delta_n \cup \{\beta\} \\ \Theta_n + 1 = \Theta_n \end{array}$$

In any case $(\Delta_{n+1}, \Theta_{n+1})$ is CD.

Now let $(\Delta, \Theta) = \bigcup_n(\Delta_n, \Theta_n)$, this theory is the saturated theory.

Definition 15. *Let* S *be the set of all complete theories in the predicate language FSL. If the logic is CD then all the theories are in the language with variables $\mathcal{V}$, if the logic is not CD, then assume that each theory leaves us an infinite number of variables from $\mathcal{V}$ not in the theory. We can write (Δ, Θ) as Δ because for a saturated theory (Δ, Θ), we have $\Theta = \{\beta | \Delta \nvdash \beta\}$.*

Define two relations on S

1. *(set inclusion) $\Delta \subseteq \Delta^{'}$*
2. *For every $\{x\}\varphi(x)$ let $\Delta R_{\{x\}\varphi(x)} \Delta^{'}$ iff for all ψ such that $\{x\}\varphi$ **says** $\psi \in \Delta$ we have $\psi \in \Delta^{'}$.*

Lemma 4. *Suppose $\Delta \nvdash \alpha \to \beta$, then for some $\Delta^{'} \supseteq \Delta$, $\Delta^{'} \vdash \alpha$ and $\Delta' \nvdash \beta$*

Proof. *From hypothesis we have*

$$\Delta \cup \{\alpha\} \nvdash \beta$$

and $\Delta \cup \{\alpha\}$ can be completed to be a saturated theory $\Delta^{'}$ such that

$$\Delta^{'} \nvdash \beta$$

In case of logic CD, this can be done in the same language with variables $\mathcal{V}$. If the logic is not CD, then since there is an infinite number of variables not in Δ, $\Delta^{'}$ can use some of them, still leaving infinitely out of Δ

Lemma 5. *Assume $\Delta \nvdash \forall x \varphi(x)$, if the logic is not CD, then for some u not in the language od Δ, we have $\Delta \nvdash \varphi(x)$. Δ can be extended in a saturated $\Delta^{'}$ by adding the variable u and more variables such that $\Delta^{'} \nvdash \varphi(u)$, and still infinitely numbers of variables are not in $\Delta^{'}$. If the logic is CD, such a u is in the logic in Δ and $(\Delta, \{\varphi(u)\})$ can be extended to a complete and saturated theory in the same language.*

Lemma 6. *Let (Δ, Θ) be complete and saturated. Assume $\{x\}\varphi$ **says** ψ is not in Θ. Then*

$$\Delta_0 = \{\alpha | \{x\}\varphi(x) \,\textbf{says}\, \alpha \in \Delta\}$$

does not prove ψ, otherwise

$$\vdash \bigwedge \alpha_j \to \psi$$

hence

$$\vdash \bigwedge_j \{x\}\varphi(x) \,\textbf{says}\, \alpha_i \to \{x\}\varphi(x) \,\textbf{says}\, \psi$$

hence

$$\{x\}\varphi(x) \,\textbf{says}\, \in \Delta$$

Since Δ_0 does not prove ψ, and $(\Delta_0, \{\psi\})$ is consistent, we can extend Δ_0 to a saturated theory $(\Delta^{'}, \Theta^{'})$. In case the logic is CD, $(\Delta^{'}, \Theta^{'})$ will be in the same language. Otherwise we use more variables.

Lemma 7. *Properties of the model* $(S, \subseteq, R_{\{x\}\varphi})$*:*

1. $\Delta_1 \subseteq \Delta_2$ *and* $\Delta_2 R_{\{x\}\varphi} \Theta$ *then* $\Delta_1 R_{\{x\}\varphi} \Theta$

   Proof. $\Delta_2 R_{\{x\}\varphi} \Theta$ *means for every* $\{x\}\varphi \,\textbf{says}\, \psi \in \Delta_2$ *we have* $\psi \in \Theta$*. Since* $\Delta_1 \subseteq \Delta_2$ *we have for every* $\{x\}\varphi \,\textbf{says}\, \psi \in \Delta$ *we have* $\psi \in \Theta$*.*

2. *If we add the axiom* $\forall x(\varphi(x) \leftrightarrow \varphi^{'}(x)) \to (\{x\}\varphi \,\textbf{says}\, \psi \leftrightarrow \{x\}\varphi^{'} \,\textbf{says}\, \psi)$ *we get the condition*

   $$\Delta \vdash \forall x(\varphi(x) \leftrightarrow \varphi^{'}(x))$$

   implies for all Θ

   $$\Delta R_{\{x\}\varphi} \Theta \leftrightarrow \Delta R_{\{x\}\varphi'} \Theta$$

Definition 16 (Construction of the model). *Take* $(S, \subseteq, R_{\{x\}\varphi(x)})$ *as defined above. For atomic* $P(x_1, \ldots, x_n)$ *and* $\Delta \in S$*, let*

$$\Delta \models P \text{ iff } P \in \Delta$$

The domain of Δ *is defined by the variables of* Δ*. If the logic is CD all* Δ *will have variables* $\mathcal{V}$ *as domain, otherwise we will have variable domains.*

Lemma 8. *For any* ψ, Δ

$$\Delta \models \psi \text{ iff } \psi \in \Delta$$

Proof. Proof by taking in exam "$\to$*" and "***says***".*