On Formal Specification and Analysis of Security Policies

Tony Bourdier, Horatiu Cirstea, Mathieu Jaume, Hélène Kirchner

RESEARCH REPORT INRIA-00429240

On Formal Specification and Analysis of Security Policies*

Tony Bourdier[1], Horatiu Cirstea[1], Mathieu Jaume[2], Hélène Kirchner[3]

[1] INRIA Nancy Grand Est · Nancy Université · LORIA
BP 101, 54602 Villers-lès-Nancy Cedex, France
Tel.: +33354958415
{Tony.Bourdier, Horatiu.Cirstea}@loria.fr

[2] SPI - LIP6, Université Paris 6
104 av du Président Kennedy, 75016 Paris, France
Mathieu.Jaume@lip6.fr

[3] INRIA Bordeaux Sud-Ouest
351, Cours de la Libération, 33405 Talence Cedex, France
Helene.Kirchner@inria.fr

ABSTRACT. Security policies are ubiquitous in information systems and more generally in the management of sensitive information. Access control policies are probably the most largely used policies but their application goes well beyond this application domain. The enforcement of security policies is useless if some of their key properties like the consistency, for example, cannot be stated and checked. We propose here a framework where the security policies and the systems they are applied on, are specified separately but using a common formalism. This separation allows us not only some analysis of the policy independently of the target system but also the application of a given policy on different systems. Besides the abstract formalism we also explore how rewrite and reduction systems can be used and combined in a rather systematic way to provide executable specifications for this framework. We also propose a notion of system and policy transformation that gives the possibility to study some properties which cannot be expressed only within the initial presentation. We have shown, in particular, how confidentiality, integrity and confinment can be expressed for the BLP policy that does not deal explicitly with information flows but only with objects containing tractable information.

1 Introduction

When addressing the field of security policies in computer science, we are faced to multiple definitions of this concept, most often based on their purpose rather than on their behavior. For instance, in a very generic way, one can say that the purpose of a security policy is to define what it means to be secure for a system, an organization or another entity. In information systems, a security policy can put constraints on functions and information flows, on access to resources or data by external systems or persons. One may call security policies special programs that deliver authorizations to perform specific actions: for instance, they decide whether or not an access is granted, whether or not a transaction may be approved, possibly taking into account the history of transactions (e.g., on

*This work was partially supported by ANR project "SSURF". The first author has obtained a partial PhD grant from the Région Lorraine.

a bank account, the total amount of cash withdrawal during the month should not exceed a fixed amount), or priority considerations (e.g., an emergency call is always given priority). The additional specificity of such programs is their reactive behaviour with respect to their execution environment: on one hand, a running program may query the policy for an authorization before performing specific accesses or transactions; on the other hand, the answers of the policy may change the execution of the calling program by excluding possible executions that do not comply with the policy.

Most of the first general and simple policies for access control were implemented using access control matrices but, nowadays, we are confronted with a lot of specific policies whose aims can be very different. In large systems, there are many classes of subjects with different needs for processing a variety of resources. Different subjects usually have different (even competing) requirements on the use of resources and their security goals (confidentiality, availability, integrity) may be distinct. Hence, various access requirements have to be consistently authorized and maintained. Beyond access control, security policies also address complex authorizations with delegations that may involve quite intricated computations. The security policies have to be deployed in the actual context of distributed organizations, mobile users and on-line services, and this implies that these policies should be easily composable, highly reconfigurable, time dependent and reactive to their environment.

Security policies may be written by different parties at different times, so it is necessary to detect and resolve conflicts among them. Detecting and solving conflicts is crucial for the policy designers who need analysis and verification tools, as well as methodology for conflict resolution. Suitable properties of policies, such as consistency, have to be clearly identified and proved. This is impossible without a formal semantics in which to express the policies and their execution environment. Formal methods and logical approaches provide already some help but should be better tuned to the specific domain and properties of security policies. This is especially important for the formalization and understanding of specific properties such as privacy or trust. Indeed, the increasing complexity of policies raises up the complexity of techniques used to reason about them.

Faced to this increasingly complex situation, our first contribution is to propose here a framework where the security policies and the systems they are applied on, are specified separately but using a common formalism. This separation allows not only the analysis of the policy independently of the target system, but also the application of a given policy on different systems. Our second contribution is to explore how rewrite and reduction systems can be used and combined in a rather systematic way to provide executable specifications for this framework. Finally, we introduce a notion of transformation on environments that can be used to reason about properties a policy is supposed to ensure and that can be naturally expressed in a different presentation. Typically, this is the case for flow properties, such as confidentiality, integrity or confinment, which can be expressed in a generic way and checked on concrete systems secure *w.r.t.* an access control policy.

The paper is organized as follows. After Section 2 that introduces the basic concepts and notations, Section 3 defines specifications (Section 3.1), environments (Section 3.2), systems (Section 3.3), security policies (Section 3.4); all these notions allow us to define (in Section 3.5) the application of a policy to a system. Section 4 defines environment transformations and show how to use them to check security properties. Section 5 concludes with rela<ted works and further research directions.

2 Preliminaries

We give in this section some definitions and notations that we use in the following. More details can be found, for example, in [EM85, End72]. A many-sorted signature Σ is given by a set of sorts $\mathscr{S}$, a set of function symbols $\mathscr{F}$ and a set of predicate symbols $\mathscr{P}$. A function symbol f with arity $w = s_1,\ldots,s_n \in \mathscr{S}^*$ and co-arity s is written $f : w \mapsto s$. A predicate symbol p with arity $s_1,\ldots,s_n \in \mathscr{S}^*$ is written $p : w$. Variables are also sorted and $x : s$ means that variable x has sort s. The set $\mathscr{X}_s$ denotes a set of variables of sort s and $\mathscr{X} = \bigcup_{s\in S} \mathscr{X}_s$ is the set of many-sorted variables. Many-sorted terms are built on many-sorted signatures and classified according to their sorts. The set of terms of sort s, denoted $\mathscr{T}^s_{\Sigma,\mathscr{X}}$ is the smallest set containing $\mathscr{X}_s$ and all the terms $f(t_1,\ldots,t_n)$ such that $f : s_1,\ldots,s_n \mapsto s$ and $t_i \in \mathscr{T}^{s_i}_{\Sigma,\mathscr{X}}$ for $i \in [1..n]$. The set of all sorted terms is $\mathscr{T}_{\Sigma,\mathscr{X}} = \bigcup_{s\in\mathscr{S}} \mathscr{T}^s_{\Sigma,\mathscr{X}}$. Let $\mathbb{N}_+$ be the set of positive naturals and $\mathbb{N}^*_+$ the corresponding monoid with neutral element ε and the concatenation operator ".". We call position any element ω of $\mathbb{N}^*_+$. For all $m,n \in \mathbb{N}^*_+$, m is a prefix of n, denoted by $m \leq n$, if it exists $n' \in \mathbb{N}^*_+$ such that $n = m.n'$. A term (seen as a tree) is an application t from a non-empty part $\mathscr{P}os(t)$ of $\mathbb{N}^*_+$ to $\mathscr{F} \cup \mathscr{X}$ such that $\mathscr{P}os(t)$ is closed under prefix (*i.e.* if $\omega \in \mathscr{P}os(t)$, then all prefixes of ω belong to $\mathscr{P}os(t)$) and for all $m \in \mathscr{P}os(t)$ and any $i \in \mathbb{N}_+$, $m.i \in \mathscr{P}os(t)$ if and only if $t(m) = f \in \mathscr{F}$ and $1 \leq i \leq arity(f)$. $\mathscr{P}os(t)$ is called the set of positions of t. The notation $t_{|\omega}$ is used to denote the subterm of t at position ω. We denote by $t[s]_\omega$ the term t with the subterm at position ω replaced by s. The set of variables occurring in a term t is denoted by $\mathscr{V}ar(t)$. If $\mathscr{V}ar(t)$ is empty, t is called a ground term and $\mathscr{T}_\Sigma$ is the set of ground terms. We call substitution any mapping from $\mathscr{X}$ to $\mathscr{T}_{\Sigma,\mathscr{X}}$ which is the identity except over a finite set of variables called domain of σ and denoted by $\mathscr{D}om(\sigma)$. The set $\mathscr{C}odom(\sigma) = \{t \in \mathscr{T}_\Sigma \mid \exists x, \sigma(x) = t\}$ is called codomain of σ. Moreover, σ is extended to an endomorphism of $\mathscr{T}_{\Sigma,\mathscr{X}}$. If $\mathscr{C}odom(\sigma) \subseteq \mathscr{T}_\Sigma$, σ is said to be a ground substitution.

A Σ-atom is either an equality $s = t$ where s and t are two terms of the same sort, or an object of the form $p(t_1,\ldots,t_n)$ where $p : s_1,\ldots,s_n \in \mathscr{P}$ and $t_i \in \mathscr{T}^{s_i}_{\Sigma,\mathscr{X}}$ for $i \in [1..n]$. The set of all Σ-atoms is denoted by $\mathscr{A}t_\Sigma$. A Σ-litteral if either a Σ-atom (positive litteral) or $\neg a$ where a is a Σ-atom (negative litteral). A Σ-clause is either a litteral or $l \vee c$ where l is a litteral and c a clause. If a clause contains at most one positive litteral, it is called a Horn clause; if it contains exactly one positive litteral, it is called a definite Horn clause. The set of Σ-formulae is the smallest set of expressions, denoted by $\mathscr{F}or_\Sigma$, containing $\mathscr{A}t_\Sigma$ and such that if $\varphi, \psi \in \mathscr{F}or_\Sigma$ and $x \in \mathscr{X}$ then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $(\forall x)\varphi$, $(\exists x)\varphi$ and $(\exists! x)\varphi$ are in $\mathscr{F}or_\Sigma$. $\varphi \Rightarrow \psi$ is a shortcut for $\neg\varphi \vee \psi$. Free and bound variables of a formula φ are defined as usual in first-order logic and are respectively denoted by $\mathscr{F}Var(\varphi)$ and $\mathscr{B}Var(\varphi)$. A formula without free variables is said closed and one without any variable is said ground. We call theory any set of formulae $\mathscr{T}h$. A Horn (resp. definite Horn) theory is a theory containing only Horn (resp. definite Horn) clauses.

DEFINITION 1. *[Σ-algebra] Given a signature $\Sigma = (\mathscr{S},\mathscr{F},\mathscr{P})$, a* **$\Sigma$-algebra** *$\mathscr{A}$ is given by :*

- *for all sort $s \in \mathscr{S}$, a set $\mathscr{A}^s$ called domain of sort s. We denote by $|\mathscr{A}| = \bigcup_{s\in\mathscr{S}} \mathscr{A}^s$*
- *for each $f : s_1,\ldots,s_n \mapsto s \in \mathscr{F}$, a mapping $f_\mathscr{A} : \mathscr{A}^{s_1} \times \ldots \times \mathscr{A}^{s_n} \to \mathscr{A}^s$*
- *for each $p : s_1,\ldots,s_n \in \mathscr{P}$, a relation $p_\mathscr{A}$ over $\mathscr{A}^{s_1} \times \ldots \times \mathscr{A}^{s_n}$*

We denote by $\mathscr{A}lg_\Sigma$ the set of all Σ-algebras.

DEFINITION 2. *[Valuation, semantics, model, $\mathscr{A}$-solutions] Given a signature $\Sigma = (\mathscr{S},\mathscr{F},\mathscr{P})$, a Σ-algebra $\mathscr{A}$ and a set of variables $\mathscr{X}$ sorted by $\mathscr{S}$, an* **$\mathscr{A}$-valuation** *α is a mapping which associates*

to each variable $x : s$ an element of $\mathscr{A}^s$ and is extended to a mapping α^* from $\mathscr{T}_{\Sigma,\mathscr{X}}$ to $|\mathscr{A}|$. In order to simplify notations, α^* will be denoted by α. When $|\mathscr{A}| \subseteq \mathscr{T}_\Sigma$, an $\mathscr{A}$-valuation is a ground substitution. The semantics of a Σ-formula φ in $\mathscr{A}$ according to the $\mathscr{A}$-valuation α, denoted by $[\![\varphi]\!]^\alpha_\mathscr{A}$, is defined as usual in first-order logic. $\mathscr{A}$ is a ***model*** *of φ or φ is* ***valid in*** *$\mathscr{A}$ and we write $\mathscr{A} \models \varphi$ iff $[\![\varphi]\!]^\alpha_\mathscr{A}$ is true for any valuation α. The set of $\mathscr{A}$-valuations α such that $[\![\varphi]\!]^\alpha_\mathscr{A}$ is true is called* ***$\mathscr{A}$-solutions of*** *φ and is denoted by $\mathscr{S}ol_\mathscr{A}(\varphi)$. The set of all models of φ is denoted by $\mathscr{M}od(\varphi)$ and for any theory $\mathscr{T}h$, $\mathscr{M}od(\mathscr{T}h)$ is the set of algebras which are model of all formulae in $\mathscr{T}h$. Moreover we say that $\mathscr{T}h$ is a model of a formula φ and we write $\mathscr{T}h \models \varphi$ iff $\forall \mathscr{A} \in \mathscr{M}od(\mathscr{T}h)$, $\mathscr{A} \models \varphi$.*

DEFINITION 3. *[TRS, CTRS, UCTRS] Given a signature Σ and a countable set of variables $\mathscr{X}$, we call* ***rewrite rule*** *over Σ any pair $(lhs, rhs) \subseteq \mathscr{T}_{\Sigma,\mathscr{X}} \times \mathscr{T}_{\Sigma,\mathscr{X}}$ such that $\mathscr{V}ar(rhs) \subseteq \mathscr{V}ar(lhs)$ and $lhs \notin \mathscr{X}$. A* ***term rewriting system*** *or* ***TRS*** *over Σ is a set of rewrite rules over Σ. Given a TRS $\mathscr{R}$ and a term $t \in \mathscr{T}_\Sigma$, t rewrites to $u \in \mathscr{T}_\Sigma$ with $\mathscr{R}$, denoted by $t \to_\mathscr{R} u$, iff there exists a position ω in the term t and a ground substitution σ such that $\sigma(lhs) = t_{|\omega}$ and $u = t[\sigma(rhs)]_\omega$ (in such a case we also say that t rewrites to u with the rule $(lhs \to rhs)$).*

We call ***constrained rule*** *any tuple $(lhs, \varphi, rhs) \subseteq \mathscr{T}_{\Sigma,\mathscr{X}} \times \mathscr{F}or_\Sigma \times \mathscr{T}_{\Sigma,\mathscr{X}}$ such that $\mathscr{V}ar(rhs) \subseteq \mathscr{V}ar(lhs) \cup \mathscr{F}Var(\varphi)$ and $lhs \notin \mathscr{X}$. It is denoted $lhs \xrightarrow{\varphi} rhs$. A* ***constrained term rewriting system*** *or* ***CTRS*** *is a set of* ***constrained rules****. Given a CTRS $\mathscr{R}$, a term $t \in \mathscr{T}_\Sigma$ and a Σ-algebra $\mathscr{A}$ whose domain is a subset of $\mathscr{T}_\Sigma$, we say that t rewrites to $u \in \mathscr{T}_\Sigma$ by $\mathscr{R}$ in $\mathscr{A}$, which is also denoted by $t \to^\mathscr{A}_\mathscr{R} u$, iff there exist a rule $lhs \xrightarrow{\varphi} rhs$ in $\mathscr{R}$, a position $\omega \in \mathscr{P}os(t)$ and a ground substitution σ such that $\sigma(lhs) = t_{|\omega}$, $\mathscr{A} \models \sigma(\varphi)$ and $u = t[\sigma(rhs)]_\omega$. Indeed, this definition requires that $\mathscr{S}ol_\mathscr{A}(\varphi)$ is computable.*

As usual, given a relation $\to$, $\xrightarrow{+}$ (resp. $\xrightarrow{*}$, $\leftrightarrow$) denotes the transitive (resp. reflexive and transitive, symmetric) closure of $\to$.

Given a constrained term rewriting system defined on a signature Σ, we say that $f \in \mathscr{F}$ depends on $g \in \mathscr{F}$ iff there exists a rule $lhs \xrightarrow{\varphi} rhs$ such that $lhs(\varepsilon) = f$ and g occurs in φ. In order to avoid some recursive definitions that could lead to non terminating function evaluation, we assume that the dependency graph of functions has no loop. In such cases, the **CTRS** $\mathscr{R}$ is said to be an **unrecursive constrained term rewriting system** (**UCTRS**).

DEFINITION 4. *[Confluence, normalization, convergence, normal form] Given a signature Σ and a TRS $\mathscr{R}$, we say that a term $t \in \mathscr{T}_\Sigma$ is in normal form iff there exists no $t' \in \mathscr{T}_\Sigma$ such that $t \to_\mathscr{R} t'$. We also say that $\mathscr{R}$ is* ***confluent*** *iff for all $t, u, v \in \mathscr{T}_\Sigma$ such that $t \xrightarrow{*}_\mathscr{R} u$ and $t \xrightarrow{*}_\mathscr{R} v$, there exists t' such that $u \xrightarrow{*}_\mathscr{R} t'$ and $v \xrightarrow{*}_\mathscr{R} t'$,* ***terminating*** *iff there is no infinite chain $t_0 \to_\mathscr{R} t_1 \to_\mathscr{R} t_2 \to_\mathscr{R} \ldots$ and* ***convergent*** *iff it is confluent and terminating. In the latter case, any $t \in \mathscr{T}_\Sigma$ can be associated to a unique $t' \in \mathscr{T}_\Sigma$ in normal form such that $t \xrightarrow{*}_\mathscr{R} t'$: we say that t' is the* ***$\mathscr{R}$-normal form*** *of t and we denote it by $t' = t\downarrow_\mathscr{R}$. All these notions are extended to CTRS by replacing $\to_\mathscr{R}$ by $\to^\mathscr{A}_\mathscr{R}$ for any algebra $\mathscr{A}$. In this case, we talk about $\mathscr{A}$-confluence, $\mathscr{A}$-termination and $\mathscr{A}$-convergence. For any theory $\mathscr{T}h$, $\mathscr{R}$ is said $\mathscr{T}h$-confluent, $\mathscr{T}h$-terminating and $\mathscr{T}h$-convergent iff it is $\mathscr{A}$-confluent, $\mathscr{A}$-terminating and $\mathscr{A}$-convergent for any $\mathscr{A} \in \mathscr{M}od(\mathscr{T}h)$.*

Definitions and properties of constrained rewrite systems have been studied, for instance, in [KKR90].

We may assume given a partial order $>$ on rules of a TRS or CTRS $\mathscr{R}$ to express reduction priority. In this case, a term t rewrites to u with a rule r only if it cannot be rewritten with another rule r' such that $r' < r$. In what follows we always consider that the reductions *w.r.t.* TRSs and CTRSs are ordered by the order of presentation of rules. This gives the possibility to write a last rule with a variable as left-hand side, to handle all cases that do not match the previous patterns.

3 Security Policies over Systems

We propose here a framework where the security policies and the systems they are applied on are specified separately but using a common formalism.

We are interested in systems that change their state according to some sensitive actions. Since the notions of state and (restricted) transition are central, we first define transition systems on which policies can apply. We start from the classical notion of **labelled transition systems** (LTS) defined as tuples $(\mathscr{E}nv, \mathscr{E}nv_0, \mathscr{L}, \delta)$ where $\mathscr{E}nv$ is a set of states, $\mathscr{E}nv_0 \subseteq \mathscr{E}nv$ is a set of initial states, $\mathscr{L}$ is a set of action labels and $\delta \subseteq \mathscr{E}nv \times \mathscr{L} \times \mathscr{E}nv$ is a transition relation, and we make precise in the next sections the way these components are specified.

Since we are mainly interested in the behaviour of a system under a given policy, we focus on the sensitive actions concerned by the policy. When a (security) request is performed on a system in a given state, the new state of the system depends on the decision taken by the policy for the respective request. We can already point out that in the general case, the response to a request is not binary and the corresponding decision can be different from the usual accept/deny answer. The actions we are interested in are thus pairs query-decision that we call in what follows events.

A security policy depends partly on the information that characterize the states of a system but does not depend on the way the system evolves. We focus on this sensitive information and on all information that are relevant for the evolution of a system under a security policy and we group all these information in so-called environments that correspond to (part of) the states of a LTS. The systems are then characterized by transitions between environments and the security policies are specified as decisions for requests performed into an environment.

In this section, we first propose formal definitions for environments and events and, based on these definitions, we define systems and security policies and we formalize the application of a policy to a system. After giving these definitions in the general framework of first-order logic, we identify presentations of first-order theories where rewriting techniques can be used to get more operational concepts.

3.1 Specifications

The interactions between the policy and the system is applied on are performed through some **events**, which are exactly pairs consisting of a query and the associated decision. The formal specifications we consider should thus introduce the sorts Q(uery) and D(ecision).

DEFINITION 5. *[Specification] We call* ***specification*** *a 3-tuple* $\mathbb{SP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$ *where* Σ, Σ_{Ev} *are signatures such that the co-arity of every symbol in the event signature* Σ_{Ev} *is one of the two* Σ_{Ev}*-sorts* Q *or* D*,* $\mathbb{T}$ *is a set of* Σ*-formulae called* ***theory*** *of* $\mathbb{SP}$ *and* Δ *is a finite subset of* $\mathscr{T}_\Sigma$ *called* ***domain*** *of* $\mathbb{SP}$*. We denote by* $\mathscr{Q}$ *and* $\mathscr{D}$ *the sets* $\mathscr{T}^Q_{\Sigma \cup \Sigma_{Ev}}$ *(of queries) and* $\mathscr{T}^D_{\Sigma \cup \Sigma_{Ev}}$ *(of decisions) respectively.*

The previous definition differs from the standard notion of specification by the additional component Δ whose purpose is to restrict the domain of the algebraic models of the specification, as follows:

An algebra $\mathscr{A}$ is a model of a specification $\mathbb{SP} = (\Sigma, \mathbb{T}, \Delta)$ iff $\mathscr{A} \in \mathscr{A}lg_\Sigma$ such that $|\mathscr{A}| = \Delta$ and is a model of each formula in $\mathbb{T}$. We denote by $\mathscr{M}od(\mathbb{SP})$ the set of all models of $\mathbb{SP}$.

EXAMPLE 6. *Along the lines of this paper, we consider the mandatory part of the Bell and LaPadula (BLP) access control policy [BL96a, BL73]. The BLP policy constrains accesses done by subjects (S) over objects (O) according to access modes (A) by considering levels of security (belonging to a finite lattice (L, inf)) associated with subjects and objects. Hence, we define the two following signatures:*

- $\Sigma_{BLP} = (\mathscr{S}_{BLP}, \mathscr{F}_{BLP}, \mathscr{P}_{BLP})$ *where:*

$$\mathscr{S}_{BLP} = \{S, O, A, L\} \quad \mathscr{F}_{BLP} = \left\{ \begin{array}{lcll} r & : & & \mapsto A \\ w & : & & \mapsto A \\ f_s & : & S & \mapsto L \\ f_o & : & O & \mapsto L \end{array} \right\} \quad \mathscr{P}_{BLP} = \left\{ \begin{array}{lcl} inf & : & L, L \\ m & : & S, O, A \end{array} \right\}$$

  The functions f_s and f_o describe security levels associated with subjects and objects. The predicate m describes current accesses over objects by subjects: $m(s,o,a)$ means that the subject s has an access over an object o according to the access mode a.
- $\Sigma_{Ev} = (\{Q, D\}, \mathscr{F}_{Ev}, \varnothing)$ *where:*

$$\mathscr{F}_{Ev} = \left\{ \begin{array}{llllllll} ask & : & S,O,A & \mapsto & Q & permit & : & \mapsto D \\ release & : & S,O,A & \mapsto & Q & deny & : & \mapsto D \end{array} \right\}$$

  $ask(s,o,a)$ (resp. $release(s,o,a)$) means that the subject s asks to get (resp. to release) an access over an object o according to the access mode a.
- *Δ_{BLP} contains all constants of Σ_{BLP}, namely $\{r, w\}$.*

We are now in position to define the specification $\mathbb{SP}_{BLP} = (\Sigma_{BLP} \cup \Sigma_{Ev}, \mathbb{T}_{BLP}, \Delta_{BLP})$ where $\mathbb{T}_{BLP}$ is the theory expressing that (L, inf) is a lattice :

$$\mathbb{T}_{BLP} = \left\{ \begin{array}{l} \forall x\, inf(x,x) \\ \forall x,y\, inf(x,y) \wedge inf(y,x) \Rightarrow x = y \\ \forall x,y,z\, inf(x,y) \wedge inf(y,z) \Rightarrow inf(x,z) \\ \forall x,y\, \exists! z\, (inf(z,x) \wedge inf(z,y) \wedge \forall w\, (inf(w,x) \wedge inf(w,y) \Rightarrow inf(w,z))) \\ \forall x,y\, \exists! z\, (inf(x,z) \wedge inf(y,z) \wedge \forall w\, (inf(x,w) \wedge inf(y,w) \Rightarrow inf(z,w))) \end{array} \right.$$

As illustrated in the previous examples, the specifications (of security policies) are in general defined for generic sets of subjects, objects, actions.... However, in specific environments where they are applied, it is necessary to consider specific instances of these sets. The notion of extended specification defined below provides a formal way to consider such instances.

DEFINITION 7. *[Extended specification] Let $\mathbb{SP} = (\Sigma, \mathbb{T}, \Delta)$ be a specification where $\Sigma = (\mathscr{S}, \mathscr{F}, \mathscr{P})$. For any pair $\mathscr{E} = (\mathscr{F}_\mathscr{E}, \Delta_\mathscr{E})$ where $\mathscr{F}_\mathscr{E}$ is a finite set of functional symbols whose arity and coarity are in $\mathscr{S}$ and $\Delta_\mathscr{E}$ a finite set of ground terms build from Σ and $\mathscr{F}_\mathscr{E}$, we call* ***specification extended by*** *$\mathscr{E}$ from $\mathbb{SP}$ the specification denoted by $\mathbb{SP}[\mathscr{E}]$ and defined by $(\Sigma[\mathscr{E}], \mathbb{T}, \Delta[\mathscr{E}])$ where $\Sigma[\mathscr{E}] = \{\mathscr{S}, \mathscr{F} \cup \mathscr{F}_\mathscr{E}, \mathscr{P}\}$ and $\Delta[\mathscr{E}] = \Delta \cup \Delta_\mathscr{E}$.*

EXAMPLE 8. *Extended specifications are useful to consider particular sets of constants involved in the considered policy. For example, if we want to deal with the following sets:*

$$Subj = \{Bob, Alice, Charlie : \ \mapsto S\} \qquad Obj = \{File_1, File_2, File_3 : \ \mapsto O\}$$
$$Levels = \{Secret, Confidential, L_1, L_2, Public, Sanitized : \ \mapsto L\}$$

we can consider the extended specification $\mathbb{SIP}_{BLP}[\mathscr{E}]$ *where* $\mathscr{F}_{\mathscr{E}} = \Delta_{\mathscr{E}} = Subj \cup Obj \cup Levels$.

3.2 Environments

As already mentioned, the information needed by a policy to take a decision with respect to a request or to define the evolution of a system are gathered in so-called environments. An environment can be thus viewed as (the significant fragment of) the current state of a system; it consists, for example, of the set of subjects, the set of objects and the set of current accesses for an access control policy.

It seems natural to use a first-order language for the definition of the various components of a system and of a security policy. In particular, the terms of a many-sorted signature could be (and often are) used to specify the various components of a system and of a policy applied on it. Nevertheless, this is somewhat restrictive since the different actors (like the subjects and objects in an access control policy) involved in such a specification should be made precise only when the system and policy are defined. An alternative approach consists in interpreting the symbols used in the definition of the policy into the signature that is effectively used. We define thus the environments as syntactic interpretations, while the queries and decisions (*i.e.* the actions) are simply first-order terms built on a given signature.

More precisely, we define an environment as the model of a set (*i.e.* of a conjunction) of first-order formulae called theory, which further constrains the interpretation by expressing, for instance, some static properties of the environment. For instance, in Example 6, these constraints are used to specify that the security levels of the Bell and LaPadula policy are organized in a finite lattice.

A crucial point in our approach is to be able to decide whether a given first-order formula expressed as a constraint, holds in the environment. As the information needed by a policy always depend on a finite set of entities (subjects, objects, resources and so on), we will consider for defining an environment, algebras with a finite interpretation domain composed only of ground terms. The following definition formalizes the construction of such algebras.

DEFINITION 9. *[Finite term generated* Σ*-algebra] Given a signature* $\Sigma = (\mathscr{S}, \mathscr{F}, \mathscr{P})$*, we call* ***finite term generated*** Σ***-algebra*** *any* Σ*-algebra* $\mathscr{A}$ *such that for all* $s \in \mathscr{S}$*,* $\mathscr{A}^s$ *is a finite subset of* $\mathscr{T}_\Sigma$ *sorted by* s *and such that any term* t *belonging to this subset is interpreted by itself:* $t_{\mathscr{A}} = t$*. We denote by FTG-*$\mathscr{A}lg_\Sigma$ *the set of all* ***finite term generated*** Σ*-algebras and by FTG-*$\mathscr{A}lg^\Delta_\Sigma$ *the set of all finite term generated* Σ*-algebras whose domain is* Δ*. The notation FTG-*$\mathscr{M}od(\mathscr{T}h)$ *(resp. FTG-*$\mathscr{M}od^\Delta(\mathscr{T}h)$*) refers to an FTG-*$\mathscr{A}lg_\Sigma$ *(resp. FTG-*$\mathscr{A}lg^\Delta_\Sigma$*) model of* $\mathscr{T}h$ *where* $\mathscr{T}h$ *is a* Σ*-theory.*

We are now ready to define the notions of environment and event, based on a given specification.

DEFINITION 10. *[Environment, Event] Given a specification* $\mathbb{SIP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$ *(extended or not), an* $\mathbb{SIP}$***-environment*** *e is a* Δ*-FTG* Σ*-algebra model of* $\mathbb{SIP}$ *(i.e. an element of FTG-*$\mathscr{M}od^\Delta(\mathbb{T})$*). We will denote the set of all* $\mathbb{SIP}$*-environments by* $\mathscr{E}nv_{\mathbb{SIP}}$ *(or simply* $\mathscr{E}nv$ *when there is no ambiguity). An* $\mathbb{SIP}$***-event*** *(or simply* ***event****) is a pair* $\gamma = (q, d)$ *where* $q \in \mathscr{Q}$ *is the* ***query*** *of* γ *and* $d \in \mathscr{D}$ *its*

decision. *The set of all* $\mathbb{SP}$*-events is denoted* $\mathscr{L}_{\mathbb{SP}}$ *(the notation anticipates the use of events as labels in transition systems).*

EXAMPLE 11. *If we consider the* $\mathbb{SP}_{BLP}[\mathscr{E}]$ *specification of Example 8, the following algebra, denoted by* e_{BLP}*, is an environment in* $\mathscr{E}nv_{\mathbb{SP}_{BLP}[\mathscr{E}]}$*:*

- $(e_{BLP})^S = Subj \qquad (e_{BLP})^O = Obj \qquad (e_{BLP})^L = Levels$
- $(f_s)_{e_{BLP}} : \left\{ \begin{array}{lcl} Bob & \mapsto & Secret \\ Alice & \mapsto & L_2 \\ Charlie & \mapsto & Sanitized \end{array} \right. \qquad (f_o)_{e_{BLP}} : \left\{ \begin{array}{lcl} File_1 & \mapsto & Confidential \\ File_2 & \mapsto & L_1 \\ File_3 & \mapsto & L_2 \end{array} \right.$
- $inf_{e_{BLP}} = \left\{ \begin{array}{l} (Secret,Secret),(Confidential,Confidential),(Confidential,Secret), \\ (L_1,L_1),(L_1,Confidential),(L_1,Secret),(L_2,L_2),(L_2,Confidential), \\ (L_2,Secret),(Public,Public),(Public,L_1),(Public,L_2), \\ (Public,Confidential),(Public,Secret),(Sanitized,Sanitized), \\ (Sanitized,Public),(Sanitized,L_1),(Sanitized,L_2), \\ (Sanitized,Confidential),(Sanitized,Secret) \end{array} \right\}$
- $m_{e_{BLP}} = \varnothing$

Inspired by the construction of a minimal Herbrand model in first-order logic, we now define a minimal finite term generated model and give conditions under which such a minimal model exists. This definition also allows us to characterize any environment e by a theory $\mathscr{T}h$ such that e is a minimal finite term generated model of $\mathscr{T}h$. When an algebra satisfies an infinite set of ground atoms, the interest of such a characterization is obvious; when the set of ground atoms valid in the algebra is finite, then such a characterization allows us to define environments in a more "compact" way and gives the possibility to have more efficient algorithms to compute solutions of a formula in an environment. Moreover, we need a concrete way to express environments in order to compare or modify them.

DEFINITION 12.*[Minimal* Δ*-FTG model] Given a signature* $\Sigma = (\mathscr{S},\mathscr{F},\mathscr{P})$*, a subset* Δ *of* $\mathscr{T}_\Sigma$ *and a consistent definite Horn* Σ*-theory* $\mathscr{T}h$*, we call* ***minimal*** Δ***-FTG model*** *of* $\mathscr{T}h$ *the* Δ*-FTG algebra model of* $\mathscr{T}h$*, denoted by FTG-*$\mathscr{M}od^{\Delta}_{min}(\mathscr{T}h)$*, such that there is no other* Δ*-FTG model that satisfies only a strict subset of the ground atoms satisfied by FTG-*$\mathscr{M}od^{\Delta}_{min}(\mathscr{T}h)$*.*

Such a minimal model exists if for any $t \in \mathscr{T}_\Sigma$, there exists $t' \in \Delta$ such that $\mathscr{T}h \models t = t'$. Indeed, if for any $t \in \mathscr{T}_\Sigma$, there exists $t' \in \Delta$ such that $\mathscr{T}h \models t = t'$, then there exists at least one Δ-*FTG* algebra model of $\mathscr{T}h$. Since each Δ-*FTG* algebra has a finite domain of cardinality $card(\Delta)$, there is a finite number of ground atoms valid in this algebra (we only consider ground atoms built with terms in Δ knowing that if an atom contains a term $t \notin \Delta$, it suffices to replace t by the term t' such that $\mathscr{T}h \models t = t'$). More precisely, there exist $card(\Delta) \times (card(\Delta)-1) + \sum_{p\in\mathscr{P}} card(\Delta)^{ar(p)}$ Δ-*FTG* models of $\mathscr{T}h$. Thus, by representing each finite term generated model by the set of ground atoms it satisfies, it is easy to see that $\mathscr{A}_{min} = \bigcap_{\mathscr{A}\in FTG\text{-}\mathscr{M}od^{\Delta}_{\Sigma}(\mathbb{T})} \mathscr{A}$ is a Δ-*FTG* algebra and that it is a model of $\mathscr{T}h$, since $\mathscr{T}h$ contains only definite Horn clauses. Such $\mathscr{A}_{min}$ is the minimal model *FTG-*$\mathscr{M}od^{\Delta}_{min}(\mathscr{T}h)$.

DEFINITION 13.*[Presentation of a specification, of an environment] Given a specification* $\mathbb{SP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$*, we call* ***presentation*** *of* $\mathbb{SP}$ *any tuple* $\langle\mathbb{SP}\rangle = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}_{cons} \cup \mathbb{T}_{ded}, \Delta)$ *such that*

- $\mathbb{T}_{cons}$ *is a set of definite Horn clauses containing only atoms of the form* $p(t_1,\ldots,t_n)$ *where for all* $i \in [1..n]$*,* t_i *is either a variable or a ground term,*

- $\mathbb{T}_{cons} \cup \mathbb{T}_{ded}$ *is semantically equivalent to* $\mathbb{T}$.

Moreover, given a presentation $\langle \mathbb{SP} \rangle$ *of* $\mathbb{SP}$*, we call* ***presentation*** *(w.r.t* $\langle \mathbb{SP} \rangle$*) of the* $\mathbb{SP}$*-environment* e *the pair:*

$$\langle e \rangle = (\mathscr{B}, \mathscr{R})$$

such that

- $\mathscr{B}$ *is a set of ground atoms called* ***base of facts****,*
- $\mathscr{R}$ *is a convergent UCTRS called* ***interpretation of functions*** *whose normal forms belong to* Δ*,*
- $e = FTG\text{-}\mathscr{M}od^{\Delta}_{min}(\mathscr{B} \cup \mathbb{T}_{cons} \cup \{t = t' \mid t, t' \in \mathscr{T}_{\Sigma} \text{ and } t' = t\downarrow^{e}_{\mathscr{R}}\})$

Just assuming that every function in the signature has a computable interpretation, every specification has indeed a presentation as above.

PROPOSITION 14. *[Existence of a presentation] Given a presentation* $\langle \mathbb{SP} \rangle$ *and an* $\mathbb{SP}$*-environment* e*, there always exists a presentation of* e *w.r.t* $\langle \mathbb{SP} \rangle$*, provided that for any* $f \in \mathscr{F}$*,* f_e *is computable.*

PROOF. Since e is finite and term generated, we can build $\mathscr{B}$ as the finite set of all ground atoms valid in e in which only terms of Δ occur, and $\mathscr{R} = \{(f(t_1,\ldots,t_n) \to t) \mid f_e(t_1,\ldots,t_n) = t,\ t_1,\ldots,t_n \in \Delta\}$.

The construction given above is rather naive and inefficient in the sense that it considers explicitly every ground atom built from Δ. The aim is to take into account $\mathbb{T}_{cons}$ and to build a suitable $\mathscr{R}$ in order to obtain a smaller set $\mathscr{B}$. For example, given a relation r closed under a property (e.g. transitivity), it is more natural to express the smallest subset of r whose (transitive) closure is r as the base of facts and to add the closure property (transitivity) in $\mathbb{T}_{cons}$, instead of enumerating all elements in relation by r in the base of facts.

EXAMPLE 15. *From the specification* $\mathbb{SP}_{BLP}[\mathscr{E}]$*, we can define the presentation* $\langle \mathbb{SP}_{BLP}[\mathscr{E}] \rangle = (\Sigma_{BLP} \cup \Sigma_{Ev}, \mathbb{T}^{BLP}_{cons} \cup \mathbb{T}^{BLP}_{ded}, \Delta_{\mathscr{E}})$ *where:*

$$\mathbb{T}^{BLP}_{cons} = \left\{ \begin{array}{l} \forall x\, inf(x,x) \\ \forall x,y,z\, inf(x,y) \wedge inf(y,z) \Rightarrow inf(x,z) \end{array} \right\}$$

$$\mathbb{T}^{BLP}_{ded} = \left\{ \begin{array}{l} \forall x,y\, inf(x,y) \wedge inf(y,x) \Rightarrow x = y \\ \forall x,y \exists! z\, (inf(z,x) \wedge inf(z,y) \wedge \forall w\, (inf(w,x) \wedge inf(w,y) \Rightarrow inf(w,z))) \\ \forall x,y \exists! z\, (inf(x,z) \wedge inf(y,z) \wedge \forall w\, (inf(x,w) \wedge inf(y,w) \Rightarrow inf(z,w))) \end{array} \right\}$$

Now, if we consider the environment e_{BLP} *introduced in Example 11, its presentation can be defined as* $\langle e_{BLP} \rangle = (\mathscr{B}_{e_{BLP}}, \mathscr{R}_{e_{BLP}})$ *where:*

$$\mathscr{B}_{e_{BLP}} = \left\{ \begin{array}{l} inf(Confidential, Secret), inf(L_1, Confidential), inf(L_2, Confidential), \\ inf(Public, L_1), inf(Public, L_2), inf(Sanitized, Public) \end{array} \right\}$$

$$\mathscr{R}_{e_{BLP}} = \left\{ \begin{array}{lll} (f_s(Bob), Secret) & (f_s(Alice), L_2) & (f_s(Charlie), Sanitized) \\ (f_o(File_1), Confidential) & (f_o(File_2), L_1) & (f_o(File_3), L_2) \end{array} \right\}$$

To show the usefulness of using a UCTRS to interpret function symbols, let us assume that the security level of a subject depends on all objects it has read: for example, it is the infimum of all these

objects (provided there is at least one object). In such a case, we add to $\mathscr{R}_{e_{BLP}}$ the following rule:

$$\left(f_s(s), \varphi = \left[\begin{array}{l} \exists o, m(s,o,r) \\ \wedge \forall o, (m(s,o,r) \Rightarrow inf(l_{inf}, f_o(o))) \\ \wedge \forall l, (\forall o, m(s,o,r) \Rightarrow inf(l, f_o(o))) \Rightarrow inf(l, l_{inf})) \end{array} \right], l_{inf} \right)$$

This rule must be applied in priority, so is added as the first rule of the ordered UCTRS $\mathscr{R}_{e_{BLP}}$. To ensure that $\mathscr{R}_{e_{BLP}}$ is still convergent, it is sufficient to prove that for any $e \in \mathscr{E}nv_{\mathbb{SP}}$, the set of e-solutions of φ contains at most one element. The formula $\forall x, y \exists ! z (inf(z,x) \wedge inf(z,y) \wedge \forall w (inf(w,x) \wedge inf(w,y) \Rightarrow inf(w,z)))$ belonging to $\mathbb{T}_{ded}^{BLP}$ allow us to do such a proof.

PROPOSITION 16. *Given an $\mathbb{SP}$-environment e, if there exists a presentation $\langle e \rangle = (\mathscr{B}, \mathscr{R})$ of e w.r.t. to a presentation $\langle \mathbb{SP} \rangle = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}_{cons} \cup \mathbb{T}_{ded}, \Delta)$, then the set of all e-solutions of φ is computable for all $\varphi \in \mathscr{F}or_{\Sigma}$.*

PROOF. We give a proof by induction over $\mathscr{F}or_{\Sigma}$.

- $e \models \sigma(p(t_1, \ldots, t_n))$ iff $\sigma(p(t_1, \ldots, t_n)) \downarrow_{\mathscr{R}}^{e} \in \mathscr{B}$ or there exists $(l_1 \wedge \ldots \wedge l_n) \Rightarrow l_{n+1} \in \mathbb{T}_{cons}$ such that $e \models \sigma(l_i)$ for all $i \in [1..n]$ and $\sigma(p(t_1, \ldots, t_n)) \downarrow_{\mathscr{R}}^{e} = \sigma(l_{n+1}) \downarrow_{\mathscr{R}}^{e}$. Thus, if φ is an atom, then $\mathscr{S}ol_e(\varphi)$ is computable since $\downarrow_{\mathscr{R}}^{e}$ is convergent and the set of possible substitutions is finite (there are $card(\Delta)^{card(\mathscr{F}Var(\varphi))}$ possible substitutions).
- $e \models \sigma(\neg\varphi)$ iff $e \not\models \sigma(\varphi)$. Thus, if $\mathscr{S}ol_e(\varphi)$ is computable, then $\mathscr{S}ol_e(\neg\varphi)$ is computable too and equals to the complement of $\mathscr{S}ol_e(\varphi)$.
- if $\mathscr{S}ol_e(\varphi_1)$ and $\mathscr{S}ol_e(\varphi_2)$ are computable, then $\mathscr{S}ol_e(\varphi_1 \wedge \varphi_2)$ and $\mathscr{S}ol_e(\varphi_1 \vee \varphi_2)$ are obviously computable and respectively equal to $\mathscr{S}ol_e(\varphi_1) \cap \mathscr{S}ol_e(\varphi_2)$ and $\mathscr{S}ol_e(\varphi_1) \cup \mathscr{S}ol_e(\varphi_2)$.
- finally, we have
$$\mathscr{S}ol_e((\forall x)\varphi) = \bigcap_{t \in \Delta} \mathscr{S}ol_e(\{x \mapsto t\}(\varphi))$$
and
$$\mathscr{S}ol_e((\exists x)\varphi) = \bigcup_{t \in \Delta} \mathscr{S}ol_e(\{x \mapsto t\}(\varphi))$$

3.3 Systems

A system is defined by a specification, a set of initial environments and a transition relation that defines the way environments evolve.

DEFINITION 17. *[System] A **system** is a tuple $\mathfrak{S} = (\mathbb{SP}, \Im nit, \delta)$ where :*

- *$\mathbb{SP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$ is a specification,*
- *$\Im nit$ is a set of Σ-formulae called **initial conditions** of $\mathfrak{S}$,*
- *δ is a functional[†] and total[‡] relation over $\mathscr{E}nv_{\mathbb{SP}} \times \mathscr{L}_{\mathbb{SP}} \times \mathscr{E}nv_{\mathbb{SP}}$ called **transition**. We note $(e,(q,d),e') \in \delta, e' = \delta(e,(q,d))$ or $e \xrightarrow{(q,d)} e'$.*

In this notion of system, we capture the evolution of a set of initial environments when a query is asked and a decision is taken. The set of initial conditions $\Im nit$ induces the set $\mathscr{E}nv_0$ of initial environments which are all environments models of $\Im nit$, that is to say $\mathscr{E}nv_0 = \mathscr{E}nv_{\mathbb{SP}} \cap \mathscr{M}od(\Im nit)$.

[†] $\forall e, e', e'' \in \mathscr{E}nv_{\mathbb{SP}}, \forall (q,d) \in \mathscr{L}_{\mathbb{SP}}, [(e,(q,d),e') \in \delta \wedge (e,(q,d),e'') \in \delta] \Rightarrow e' = e''$

[‡] $\forall (e,(q,d)) \in \mathscr{E}nv_{\mathbb{SP}} \times \mathscr{L}_{\mathbb{SP}}, \exists e' \in \mathscr{E}nv_{\mathbb{SP}}, (e,(q,d),e') \in \delta$

From any initial environment $e_0 \in \mathscr{E}nv_0$, the directed labelled graph starting from e_0 and whose labelled edges are steps $e \xrightarrow{\gamma=(q,d)} e'$ represents all possible evolutions of the environment of the system. We note δ^* the natural extension of δ to $\mathscr{E}nv_{\mathbb{SP}} \times \mathscr{L}^*_{\mathbb{SP}} \times \mathscr{E}nv_{\mathbb{SP}}$. From the set $\mathscr{E}nv_0$ of all possible initial environments and a system $\mathfrak{S}$, we define the set of reachable environments as $\Gamma(\mathfrak{S}) = \{e \in \mathscr{E}nv_{\mathbb{SP}} \mid \exists (e_0, \gamma) \in \mathscr{E}nv_0 \times \mathscr{L}^*_{\mathbb{SP}}, (e_0, \gamma, e) \in \delta^*\}$.

We dispose so far of a way to express a specification and corresponding environments and we need now a way to express the transition relation. For this, we introduce a relation between algebras which can be seen as a closeness relation.

DEFINITION 18. *[Distance between environments] Given a specification $\mathbb{SP}$, we define a distance μ between $\mathbb{SP}$-environments as follows:*

$$\mu(e,e') = \{\varphi \text{ ground atom} \mid (e \models \varphi \text{ and } e' \not\models \varphi) \text{ or } (e \not\models \varphi \text{ and } e' \models \varphi)\}$$

for any $e, e' \in \mathscr{E}nv_{\mathbb{SP}}$.

Indeed, it is natural to call μ a distance because for any $e, e', e'' \in \mathscr{E}nv_{\mathbb{SP}}$: $\mu(e,e') = \mu(e',e)$, $\mu(e,e'') \subseteq \mu(e,e') \cup \mu(e',e'')$ and $\mu(e,e') = \varnothing$ iff $e = e'$.

DEFINITION 19. *[φ-closest environment] Given a specification $\mathbb{SP}$ and a formula φ, we call **set of φ-closest environments** to e the set of environments $e_{\oplus\varphi} = \{e_1, \ldots, e_n\}$ such that:*

- $\forall i \in [1..n]$, $e_i \models \varphi$
- $\forall e' \in \mathscr{E}nv_{\mathbb{SP}}$ *such that* $e' \models \varphi$, $\exists i \in [1..n]$ *such that* $\mu(e,e') \supseteq \mu(e_i,e')$

We can obviously see that if φ is semantically equivalent to a formula of the form $\varphi_{pre} \Rightarrow \varphi_{goal}$ where φ_{goal} is a literal containing no quantified variable, then there exists $e' \in e_{\oplus\varphi}$ such that e' is exactly the algebra such that:

- *if $\varphi_{goal} = p(t_1, \ldots, t_n)$, then $\forall f \in \mathscr{F}$, $f_e = f_{e'}$, $\forall q \in \mathscr{P} \setminus \{p\}$, $q_e = q_{e'}$ and $p_{e'} = p_e \cup \Big(\bigcup_{\sigma \in \mathscr{S}ol_e(\varphi_{pre})} (\sigma(t_1)_e, \ldots, \sigma($*
- *if $\varphi_{goal} = \neg p(t_1, \ldots, t_n)$, then $\forall f \in \mathscr{F}$, $f_e = f_{e'}$, $\forall q \in \mathscr{P} \setminus \{p\}$, $q_e = q_{e'}$ and $p_{e'} = p_e \setminus \Big(\bigcup_{\sigma \in \mathscr{S}ol_e(\varphi_{pre})} (\sigma(t_1)_e, \ldots, \sigma(t_n)_e)\Big)$*
- *if φ_{goal} is of the form $f(t_1, \ldots, t_n) = t$, then $\forall q \in \mathscr{P}$, $q_e = q_{e'}$; $\forall g \in \mathscr{F} \setminus \{f\}$, $g_e = g_{e'}$; $f_{e'}$ is the function which associates $\sigma(t)_e$ to $(\sigma(t_1)_e, \ldots, \sigma(t_n)_e)$ for any $\sigma \in \mathscr{S}ol_e(\varphi_{pre})$, and $f_e(t'_1, \ldots, t'_n)$ to other tuple $(t'_1, \ldots, t'_n) \in \Delta^n$.*

Such an algebra e' will be denoted by $e_{[\varphi_{pres} \Rightarrow \varphi_{goal}]}$ and is said to be the φ-closest environment to e headed for φ_{goal}.

EXAMPLE 20. *From the extended specification $\mathbb{SP}_{BLP}[\mathscr{E}]$ and an arbitrary set $\Im nit_{BLP}$ of axioms, we can define the system $\mathfrak{S}_{BLP} = (\mathbb{SP}_{BLP}[\mathscr{E}], \Im nit_{BLP}, \delta_{BLP})$ where δ_{BLP} is defined as follows:*

$$\forall e \in \mathscr{E}nv_{\mathbb{SP}}, \begin{cases} e \xrightarrow{(ask(s,o,a),permit)}_{BLP} e_{[m(s,o,a)]} \\ e \xrightarrow{(release(s,o,a),permit)}_{BLP} e_{[\neg m(s,o,a)]} \\ e \xrightarrow{(q,deny)}_{BLP} e \end{cases}$$

Note that δ_{BLP} is a functional and total relation. For $\Im nit_{BLP}$ we can take for example the following set of formulae expressing that initially there are no accesses, and specifying the lattice of security

levels:

$$\Im nit_{BLP} = \left\{ \begin{array}{l} \forall x_s, x_o, x_a \; \neg m(x_s, x_o, x_a) \\ inf(Confidential, Secret), inf(L_1, Confidential), inf(L_2, Confidential), \\ inf(Public, L_1), inf(Public, L_2), inf(Sanitized, Public) \end{array} \right\}$$

3.4 Security Policies

As we said, a policy defines the decision that answers a request performed into a given environment. These decisions depend on the corresponding request and on the environment in which the request is performed. Indeed, the environments could be rather complex; in particular the subjects and objects that are usually involved in an access control policy are part of the environment.

DEFINITION 21. *[Security policy] A* ***security policy*** *is a pair* $\wp = (\mathbb{SP}, \mathscr{R}ules)$ *consisting of a specification* $\mathbb{SP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$ *and a relation* $\mathscr{R}ules$ *over* $\mathscr{F}or_\Sigma \times \mathscr{Q} \times \mathscr{D}$.

Given a policy $\wp = (\mathbb{SP}, \mathscr{R}ules)$, for each tuple $r = (\varphi_{pre}, q, d) \in \mathscr{R}ules$, φ_{pre} is called precondition of r. For any query q, we denote by $Pre(q)$ the set

$$\{\varphi_{pre} \in \mathscr{F}or_\Sigma \mid \exists d, \; (\varphi_{pre}, q, d) \in \mathscr{R}ules\}$$

EXAMPLE 22. *The mandatory part of the BLP policy* $\wp_{BLP} = (\mathbb{SP}_{BLP}[\mathscr{E}], \mathscr{R}ules_{BLP})$ *is defined as follows:*

- *Every subject can read an object whose level of security is less than its level of security iff it does not write into an object of a lower security level:*

  $$(\psi_1, ask(s,o,read), permit) \in \mathscr{R}ules_{BLP} \text{ and}$$
  $$(\neg\psi_1, ask(s,o,read), deny) \in \mathscr{R}ules_{BLP}$$

  where $\psi_1 = inf(f_o(o), f_s(s)) \wedge [\forall o' \; m(s,o',write) \Rightarrow inf(f_o(o), f_o(o'))]$.
- *Every subject can write into an object if it does not read another object with a higher security level.*

  $$(\psi_3, ask(s,o,write), permit) \in \mathscr{R}ules_{BLP} \text{ and}$$
  $$(\neg\psi_3, ask(s,o,write), deny) \in \mathscr{R}ules_{BLP}$$

  where $\psi_3 = \forall o' \; m(s,o',read) \Rightarrow inf(f_o(o'), f_o(o))$.
- *Every subject can release any of its accesses.*

  $$(\psi_5, release(s,o,a), permit) \in \mathscr{R}ules_{BLP} \text{ and}$$
  $$(\neg\psi_5, release(s,o,a), deny) \in \mathscr{R}ules_{BLP}$$

  where $\psi_5 = m(s,o,a)$.

Independently of the system a policy is applied on, we generally require that for every query, the policy eventually gives a decision and only one.

DEFINITION 23. *A security policy $\wp = (\mathbb{SP}, \mathscr{R}ules)$ is said to be **total** iff for any $\mathbb{SP}$-environment e and any query q, there always exists at least one decision d such as there is an element (φ_{pre}, q, d) in $\mathscr{R}ules$ such that $e \models \varphi_{pre}$. This definition can be expressed as follows: $\forall q \in \mathscr{Q}_{Ev}, \mathbb{T} \models \bigvee_{\varphi \in Pre(q)} \varphi$. Moreover, $\wp$ is said **consistent** iff for any $\mathbb{SP}$-environment e and any query q, if (φ_{pre}, q, d) and (φ'_{pre}, q, d') are in $\mathscr{R}ules$ and e satisfies φ_{pre} and φ'_{pre}, then $d = d'$.*

One can notice that the conditions used in the assertions of the policy defined in Example 22 are both exhaustive (for all $q \in \mathscr{Q}_{Ev}$, $\bigvee_{\varphi \in Pre(q)} \varphi$ is a theorem, *i.e.* always valid) and exclusive (for any $q \in \mathscr{Q}_{Ev}$, $\forall \varphi \neq \varphi' \in Pre(q)$, $\{\varphi, \varphi'\}$ is contradictory); thus it can be proved that $\wp_{BLP}$ is total and consistent.

As for environments, we define now a presentation of a policy in which the evaluation of queries into decisions is performed using constraint rewriting.

DEFINITION 24. *[Presentation of a policy] Given a presentation of a specification $\langle \mathbb{SP} \rangle$, we call **presentation** of a policy $\wp = (\mathbb{SP}, \mathscr{R}ules)$ w.r.t $\langle \mathbb{SP} \rangle$ any UCTRS denoted by $\langle \wp \rangle$ such that:*

- *the set of normal forms of $\langle \wp \rangle$ is exactly $\mathscr{D}$*
- *any rule of $\langle \wp \rangle$ is of the form $lhs \xrightarrow{\varphi} rhs$ with $\varphi \in \mathscr{F}or_{\Sigma}$*
- *for any $e \in \mathscr{E}nv_{\mathbb{SP}}$, $q \rightarrow^{e}_{\langle \wp \rangle} d$ iff there exists (φ_{pre}, q, d) in $\mathscr{R}ules$ such that $e \models \varphi_{pre}$.*

EXAMPLE 25. *The presentation of the policy $\wp_{BLP} = (\mathbb{SP}_{BLP}[\mathscr{E}], \mathscr{R}ules_{BLP})$ of Example 22 w.r.t. the presentation $\langle \mathbb{SP}_{BLP}[\mathscr{E}] \rangle$ given in Example 15 can be defined as follows:*

$$\langle \wp \rangle_{BLP} = \left\{ \begin{array}{rcl} ask(s,o,r) & \xrightarrow{inf(f_o(o),f_s(s)) \wedge (\forall o')(m(s,o',w) \Rightarrow inf(f_o(o),f_o(o')))} & permit \\ ask(s,o,w) & \xrightarrow{(\forall o')(m(s,o',r) \Rightarrow inf(f_o(o'),f_o(o)))} & permit \\ release(s,o,a) & \xrightarrow{m(s,o,a)} & permit \\ x_q & \xrightarrow{\top} & deny \end{array} \right\}$$

where x_q is a variable of sort Q. This example is quite simple in that the constraint of a rewrite rule $q \xrightarrow{\varphi} d$ corresponds exactly to a precondition in $(\varphi_{pre}, q, d) \in \mathscr{R}ules$. This is not always the case. Indeed, the definition of a presentation just requires that the relation over $\mathscr{Q} \times \mathscr{D}$ induced by $\xrightarrow{*}{}^{e}_{\langle \wp \rangle}$ corresponds to the relation "$\exists \varphi \in Pre(q)$ such that $e \models \varphi$ and $(\varphi, q, d) \in \mathscr{R}ules$". In the previous example it is obvious, but we could have a more complicated situation as the following one: let us consider that a writing access is allowed only if the corresponding reading access is permitted. In such a case, we would have a rule of the form $ask(s,o,w) \xrightarrow{\top} ask(s,o,r)$.

As in [DKKS07], the properties of totality and consistency of a policy $\wp = (\mathbb{SP}, \mathscr{R}ules)$ should be expressed as properties of termination, completeness and confluence of its presentation $\langle \wp \rangle$. We leave that for further work.

3.5 Policy over system

A system only specifies what happens when an action is performed (according to a request and a decision). A policy on such a system imposes restrictions on the reachable environments, exactly as strategies impose restrictions on reachable elements in a derivation tree or a labelled transition system. Of course, a policy cannot be applied on any system but only on those whose specification

is "more general" than that of the policy. We define thus the notion of specification subsumption and use it to characterize policies compatible with a given systems.

DEFINITION 26. *Given two specifications* $\mathbb{SP} = (\Sigma \cup \Sigma_{Ev}, \mathbb{T}, \Delta)$ *and* $\mathbb{SP}' = (\Sigma' \cup \Sigma'_{Ev}, \mathbb{T}', \Delta')$*, we say that* $\mathbb{SP}$ *subsumes* $\mathbb{SP}'$ *and we note* $\mathbb{SP} \succeq \mathbb{SP}'$ *iff :*

$$\Sigma' \subseteq \Sigma, \text{ and } \Sigma'_{Ev} \subseteq \Sigma_{Ev}, \text{ and } \mathbb{T}' \subseteq \mathbb{T} \text{ (or more generally } \mathbb{T} \models \mathbb{T}'), \text{ and } \Delta' \subseteq \Delta.$$

We say that a system $\mathfrak{S} = (\mathbb{SP}, \Im nit, \delta)$ *and a policy* $\wp = (\mathbb{SP}_{\wp}, \mathscr{R}ules_{\wp})$ *are* ***compatible*** *iff* $\mathbb{SP}_{\mathfrak{S}}$ *subsumes* $\mathbb{SP}_{\wp}$*. In such a case, the* $\wp$*-secure system build from* $\mathfrak{S}$ *is obtained by restricting the transition relation of* $\mathfrak{S}$*.*

We can now define systems constrained by policies.

DEFINITION 27. *For each compatible system* $\mathfrak{S} = (\mathbb{SP}_{\mathfrak{S}}, \Im nit, \delta)$ *and policy* $\wp = (\mathbb{SP}_{\wp}, \mathscr{R}ules)$ *the* ***$\wp$-secure system build from*** $\mathfrak{S}$ *is defined as follows :*

$$\mathfrak{S}_{|\wp} = (\mathbb{SP}_{\mathfrak{S}}, \Im nit, \delta_{\wp})$$

where $\delta_{\wp}$ *is the relation containing each* $(e, \gamma, e') \in \delta$ *such that*

$$\begin{cases} \gamma = (q,d) \in \mathscr{L}_{\mathbb{SP}_{\mathfrak{S}}} \setminus \mathscr{L}_{\mathbb{SP}_{\wp}} \text{ or else} \\ \exists (\varphi_{pre}, q, d) \in \mathscr{R}ules \text{ such that } e \models \varphi_{pre} \end{cases}$$

EXAMPLE 28. *If we consider again our running example, we can define the secure system* $\mathfrak{S}_{BLP|\wp_{BLP}} = (\mathbb{SP}_{BLP}[\mathscr{E}], \Im nit_{BLP}, \delta_{\wp_{BLP}})$*.*

DEFINITION 29. *Given a system* $\mathfrak{S} = (\mathbb{SP}_{\mathfrak{S}}, \Im nit, \delta)$*, a compatible policy* $\wp = (\mathbb{SP}_{\wp}, \mathscr{R}ules)$*, and a presentation of all considered objects, we define the transition relation of the* $\wp$*-secure system build from* $\mathfrak{S}$ *as follows:* $(e,(q,d),e') \in \delta_{\wp}$ *iff* $(e,(q,d),e') \in \delta$ *and* $q \rightarrow^{e}_{\mathscr{R}'} d$ *where*[§] $\mathscr{R}' = \mathscr{R} \cup \langle \wp \rangle$ *and* $\langle e \rangle = (\mathscr{B}, \mathscr{R})$*.*

EXAMPLE 30. *Let us consider an example proposed in [SdO08, KKSdO08] and coming from the NetFilter*[¶] *documentation. More precisely, we consider a firewall whose aim is to block any traffic coming from the public network to the private network. As usual, we denote by* $ppp0$ *the interface associated to Internet connections. The private network consists of two local machines whose IP addresses are* 10.1.1.1 *and* 10.1.1.2 *and are rewritten into a same external IP address* 123.123.123.1*. This network address translation is part of the system which also contains the set of all connections established. Thus, we have the specification presentation* $\langle \mathbb{SP} \rangle = (\Sigma \cup \Sigma_{Ev}, \varnothing, \Delta)$ *with:*

$$\mathscr{F} = \begin{cases} pckt & : & Address \times Address & \rightarrow & Packet \\ eth0, ppp0 & : & & \rightarrow & Address \\ [0,255].[0,255].[0,255].[0,255] & : & & \rightarrow & Address \end{cases}$$

$$\mathscr{P} = \{ \; Established \quad : \quad Packet \qquad \mathscr{F}_{Ev} = \begin{cases} filter & : & Packet & \rightarrow & Q \\ drop, accept & : & & \rightarrow & D \end{cases}$$

[§] The set $\mathscr{R}' = \mathscr{R} \cup \langle \wp \rangle$ is ordered such that rules belonging to $\mathscr{R}$ are always "applied" before rules of $\langle \wp \rangle$.

[¶] `http://www.netfilter.org`

and Δ is the finite set of ground terms of sort Packet. The system $\mathbb{S}$ is given by $\mathbb{SP}$, the initial environment

$$\langle e_0 \rangle = \left(\varnothing, \mathscr{R}_0 = \left\{ \begin{array}{l} (pckt(10.1.1.1, ppp0), pckt(123.123.1.1, ppp0)) \\ (pckt(10.1.1.2, ppp0), pckt(123.123.1.1, ppp0)) \end{array} \right. \right)$$

consisting of the initial network address translations and specifying the fact that initially no connection is established, as well as the following transition relation:

$$\left\{ \begin{array}{lll} e & \xrightarrow{(filter(x), accept)} & e_{[Established(x)]} \\ e & \xrightarrow{(filter(x), drop)} & e \end{array} \right.$$

The following presentation represents the simple policy described above:

$$\langle \wp \rangle = \left\{ \begin{array}{ll} filter(pckt(src, dst)) & \xrightarrow{Established(pckt(src,dst))} \quad accept \\ filter(pckt(eth0, dst)) & \xrightarrow{\neg Established(pckt(eth0,dst))} \quad accept \\ filter(pckt(ppp0, dst)) & \xrightarrow{\neg Established(pckt(ppp0,dst))} \quad drop \\ filter(pckt(123.123.1.1, dst)) & \xrightarrow{\neg Established(pckt(123.123.1.1,dst))} \quad accept \\ filter(other) & \xrightarrow{\top} \quad drop \end{array} \right.$$

Thus, the $\wp$-secure system built from $\mathbb{S}$ contains the following transitions:

$$e_0 \xrightarrow{filter(pckt(10.1.1.1), ppp0), accept} e_1$$
$$e_1 \xrightarrow{filter(ppp0, pckt(192.168.2.124)), drop} e_1$$

with $\langle e_1 \rangle = (Established(pckt(10.1.1.1), ppp0), \mathscr{R}_0)$ because

$$filter(pckt(10.1.1.1), ppp0) \rightarrow^{e_0}_{\mathscr{R}_0} filter(pckt(123.123.1.1), ppp0) \rightarrow^{e_0}_{\langle \wp \rangle} accept$$

and $filter(ppp0, pckt(192.168.2.124)) \rightarrow^{e_1}_{\langle \wp \rangle} drop$.

We have shown so far how to represent in a formal way all objects related to the specification of a security policy and of the systems it can be applied on. Such a formalism allows us to reason about all these objects, but we are also interested in studying some properties which cannot be expressed only with these objects. The next section proposes a framework for performing this kind of extra reasoning.

4 Transformation of systems

A security policy is often expected to fulfill a certain security property expressed on some entities, while it is dealing with a different set of entities. A typical example is given by policies designed for ensuring flow properties: such policies do not deal with information flow but only with objects containing information to be traced. Intuitively, a link is needed between "what you do" (the policy) and "what you want" (the goal for which the policy is designed). This link is formalized in this section through a transformation of environments, whose aim is to translate an environment into another one dealing with the entities we are interested in. This transformation itself is based on the notion of morphism between two signatures.

DEFINITION 31. *[Signature morphism] A signature morphism θ from $\Sigma_1 = (\mathscr{S}_1, \mathscr{F}_1, \mathscr{P}_1)$ to $\Sigma_2 = (\mathscr{S}_2, \mathscr{F}_2, \mathscr{P}_2)$ is a couple $(\theta_{\mathscr{S}}, \theta_{\mathscr{F}})$ such that $\theta_{\mathscr{S}} : \mathscr{S}_1 \to \mathscr{S}_2$ and $\theta_{\mathscr{F}} : \mathscr{F}_1 \to \mathscr{F}_2$ are (partial or total) functions such that $\forall f : s_1, \ldots, s_n \mapsto s \in \mathscr{D}om(\theta_{\mathscr{F}})$ where $s_1, \ldots, s_n, s \in \mathscr{D}om(\theta_{\mathscr{S}})$:*

$$\theta_{\mathscr{F}}(f) : \theta_{\mathscr{S}}(s_1), \ldots, \theta_{\mathscr{S}}(s_n) \mapsto \theta_{\mathscr{S}}(s) \in \mathscr{F}_2$$

We extend θ to a mapping $\hat{\theta}$ over terms as follows:

$$\begin{array}{ll} \forall x : s \in \mathscr{X}, & \hat{\theta}(x : s) = x : \theta_{\mathscr{S}}(s) \\ \forall f \in \mathscr{D}om(\theta_{\mathscr{F}}), & \hat{\theta}(f(t_1, \ldots, t_n) : s) = \theta_{\mathscr{F}}(f)(\hat{\theta}(t_1), \ldots, \hat{\theta}(t_n)) \end{array}$$

$\hat{\theta}$ will be simply denoted by θ.

DEFINITION 32. *[Environment tranformation] Given two specifications $\mathbb{SP}_1$ and $\mathbb{SP}_2$, an environment transformation is a pair $(\theta, \rightarrowtail)$ such that:*

- *θ is a signature morphism from Σ_1 to Σ_2*
- *$\rightarrowtail \subseteq \mathscr{F}or_{\Sigma_1} \times \mathscr{F}or_{\Sigma_2}$ is a relation such that $\varphi_1 \rightarrowtail \varphi_2$ only if $\mathscr{F}Var(\varphi_1) = \mathscr{F}Var(\varphi_2)$*

An environment transformation $(\theta, \rightarrowtail)$ induces a relation $\overset{\theta}{\rightarrowtail}$ over $\mathscr{E}nv_{\mathbb{SP}_1} \times \mathscr{E}nv_{\mathbb{SP}_2}$ defined as follows:

$$e_1 \overset{\theta}{\rightarrowtail} e_2 \Leftrightarrow \left(\ \forall \varphi_1, \varphi_2 \ s.t. \ \varphi_1 \rightarrowtail \varphi_2, \ if \ \alpha \in \mathscr{S}ol_{e_1}(\varphi_1) \ then \ \theta \circ \alpha \in \mathscr{S}ol_{e_2}(\varphi_2) \ \right)$$

In order to obtain a functional relation $\overset{\theta}{\rightarrowtail}$, we adapt the previous one with respect to a presentation of specifications. Given two presentations $\langle \mathbb{SP}_1 \rangle = (\Sigma_1 \cup \Sigma_{Ev_1}, \mathbb{T}_1, \Delta_1)$ and $\langle \mathbb{SP}_2 \rangle = (\Sigma_2 \cup \Sigma_{Ev_2}, \mathbb{T}_2, \Delta_2)$ of $\mathbb{SP}_1$ and $\mathbb{SP}_2$, an environment transformation $(\theta, \rightarrowtail)$ whose image contains only definite Horn clauses, induces a relation $\overset{\theta}{\rightarrowtail}$ w.r.t. $\langle \mathbb{SP}_1 \rangle$ and $\langle \mathbb{SP}_2 \rangle$ such that for any $e_1 \in \mathscr{E}nv_{\mathbb{SP}_1}$, there exists only one $e_2 \in \mathscr{E}nv_{\mathbb{SP}_2}$ such that $e_1 \overset{\theta}{\rightarrowtail} e_2$ and which is equal to

$$FTG\text{-}\mathscr{M}od^{\Delta_2}_{min}(\mathbb{T}_{cons2} \cup \{\theta \circ \alpha(\varphi_2) \mid \varphi_1 \rightarrowtail \varphi_2 \ and \ \alpha \in \mathscr{S}ol_{e_1}(\varphi_1)\})$$

Environment transformations are useful to check the same properties over several secure systems based on different specifications. Indeed, let $\mathfrak{S} = (\mathbb{SP}, \Im nit, \delta)$ be a system based on a specification $\mathbb{SP}$ and suppose we want to prove a property Ω over reachable states of $\mathfrak{S}$ which cannot be expressed in $\mathbb{SP}$. For this, we can use another specification $\mathbb{SP}'$ in which Ω can be expressed. Then, in order to check the desired properties over reachable states of $\mathfrak{S}$, it suffices to build a transformation of $\mathbb{SP}$-environments to $\mathscr{E}nv_{\mathbb{SP}'}$ $(\theta, \rightarrowtail)$ where θ is a signature morphism from Σ to Σ' and to check that there is no reachable state $e \in \Gamma(\mathfrak{S})$ such that an $\mathbb{SP}'$-environment e' verifying $e \overset{\theta}{\rightarrowtail} e'$ satisfies $\neg\Omega$. Such a transformation is especially interesting in access control where the system and the policy manipulate subjects and objects and where the aim of the policy is to control the information flow induced by accesses. In such a case it is useful to avoid the mix between what is really manipulated (accesses to files) and what one wants to reason about (information contained in files).

EXAMPLE 33. *In this example we illustrate environment transformations in order to deal with information flow properties of access control policies. First, we introduce the "generic" signature* $\Sigma_{Flow} = (\mathscr{S}_{Flow}, \mathscr{F}_{Flow}, \mathscr{P}_{Flow})$ *where:*

$$\mathscr{S}_{Flow} = \{Actor, Information\} \quad \mathscr{P}_{Flow} = \left\{ \begin{array}{l} Get : Actor, Information, \\ Put : Actor, Information, \\ MoveTo : Information, Information \\ Eligible : Actor, Information \\ Trustworthy : Actor, Information \\ Gflow : Information, Information \end{array} \right\}$$

and where $\mathscr{F}_{Flow}$ *is an arbitrary set of function symbols.* $Get(a,i)$ *means that the actor* a *knows the information* i, $Put(a,i)$ *means that the actor* a *modifies the information* i *(by using the information he knows),* $MoveTo(i_1,i_2)$ *means that the information* i_2 *is enriched with information* i_1, $Eligible(a,i)$ *means that the actor* a *is granted to know the information* i, $Trustworthy(s,i)$ *means that the actor* a *is granted to modify the information* i *and* $Gflow(i_1,i_2)$ *means that the information* i_1 *is authorized to flow into* i_2. *The predicates* Get, Put *and* $MoveTo$ *are useful to describe flows that happen while the predicates* $Eligible$, $Trustworthy$, *and* $Gflow$ *are used to specify flow policies (respectively a confidentiality policy, an integrity policy and a confinement policy). If we consider* $\mathscr{F}_{Flow}$ *as the set* $\mathscr{F}_{\mathscr{E}}$ *used to build the extended specification* $\mathbb{SP}_{BLP}[\mathscr{E}]$, *we can define the signature morphism* $\theta^{\widetilde{BLP}} = (\theta^{\widetilde{BLP}}_{\mathscr{S}}, \theta^{\widetilde{BLP}}_{\mathscr{F}})$ *from* $\Sigma_{BLP}[\mathscr{E}]$ *to* $\Sigma_{Flow}[\mathscr{E}] = (\mathscr{S}_{Flow}, \mathscr{E}, \mathscr{P}_{Flow})$ *where:*

- $\mathscr{D}om(\theta^{\widetilde{BLP}}_{\mathscr{S}}) = \{S, O\}$ *with* $\theta^{\widetilde{BLP}}_{\mathscr{S}}(S) = Actor$ *and* $\theta^{\widetilde{BLP}}_{\mathscr{S}}(O) = Information$
- $\theta^{\widetilde{BLP}}_{\mathscr{F}}$ *is the identity function:*

$$\forall o \in Obj\ \theta^{\widetilde{BLP}}_{\mathscr{F}}(o) = o : Information \quad \forall s \in Subj\ \theta^{\widetilde{BLP}}_{\mathscr{F}}(s) = s : Actor$$

Now we introduce the presentation $\langle \mathbb{SP}_{Flow}[\mathscr{E}] \rangle = (\Sigma_{Flow}[\mathscr{E}], \mathbb{T}^{Flow}_{cons} \cup \mathbb{T}^{Flow}_{ded}, \Delta_{\mathscr{E}})$ *of a specification where:*

$$\mathbb{T}^{Flow}_{cons} = \left\{ \begin{array}{l} \forall o\ MoveTo(o,o) \\ \forall s,o,o'\ Get(s,o) \wedge Put(s,o') \Rightarrow MoveTo(o,o') \\ \forall s,o,o'\ MoveTo(o,o') \wedge Get(s,o') \Rightarrow Get(s,o) \\ \forall s,o,o' MoveTo(o,o') \wedge Put(s,o) \Rightarrow Put(s,o') \\ \forall o,o',o''\ MoveTo(o,o') \wedge MoveTo(o',o'') \Rightarrow MoveTo(o,o'') \end{array} \right\}$$
$$\mathbb{T}^{Flow}_{ded} = \varnothing$$

Here, $\mathbb{T}^{Flow}_{cons}$ *allows to consider the transitive closure of information flows. Within such a specification, it is possible to define, in a generic way, confidentiality, integrity and confinement security properties as follows:*

- *Confidentiality* ψ_{conf}: $Get(s,o) \Rightarrow Eligible(s,o)$
- *Integrity* ψ_{int}: $Put(s,o) \Rightarrow Trustworthy(s,o)$
- *Confinement* ψ_{info}: $MoveTo(o,o') \Rightarrow Gflow(o,o')$

In order to analyse information flows for the BLP policy, we define the environment transformation $(\theta^{\widetilde{BLP}}, \rightarrowtail_{\widetilde{BLP}})$ *where:*

$$\left\{ \begin{array}{llll} m(s,o,read) & \rightarrowtail_{\widetilde{BLP}} & Get(s,o) & m(s,o,write) \rightarrowtail_{\widetilde{BLP}} Put(s,o) \\ inf(f_o(o), f_s(s)) & \rightarrowtail_{\widetilde{BLP}} & Eligible(s,o) & inf(f_o(o), f_o(o')) \rightarrowtail_{\widetilde{BLP}} Gflow(o,o') \end{array} \right.$$

Such an environment transformation provides a framework allowing to check that the BLP policy ensures confidentiality and confinement. Indeed, it can be proved that:

$$\forall e \in \Gamma(\mathfrak{S}_{BLP|\wp_{BLP}}) \quad e \overset{\theta^{\widetilde{BLP}}}{\rightarrowtail}_{\widetilde{BLP}} e' \Rightarrow e' \models \psi_{conf} \wedge \psi_{info}$$

In the original paper [BL96a, BL73], the authors define the mandatory part of their policy as the two properties defined as follows:

$$\Omega_{BLP} = \left\{ \begin{array}{l} \text{MAC-property} \\ \forall o, s\, m(s,o,read) \Rightarrow inf(f_o(o), f_s(s)) \\ \\ \text{MAC}^\star\text{-property} \\ \forall o_1, o_2, s\, (m(s,o_1,read) \wedge m(s,o_2,write)) \Rightarrow inf(f_o(o_1), f_o(o_2)) \end{array} \right\}$$

It can be noticed that:

$$e \models \Omega_{BLP} \wedge e \overset{\theta^{\widetilde{BLP}}}{\rightarrowtail}_{\widetilde{BLP}} e' \Rightarrow e' \models \psi_{conf} \wedge \psi_{info}$$

Hence, ψ_{conf} and ψ_{info} can be viewed as an abstraction of MAC and MAC* that can be used to analyse other access control policies (e.g. the Chinese Wall policy). In the particular case of the BLP policy, we obtain a result similar to the well-known basic security theorem [BL96a, BL73] asserting that each environment $e \in \Gamma(\mathfrak{S}_{BLP|\wp_{BLP}})$ is such that $e \models \Omega_{BLP}$.

5 Conclusions

In this work, we have proposed an abstract formalism for the definition of systems constrained by security policies that allows a clear separation between the evolution of the system and the enforcement of the policy in the corresponding execution environment. Environments are formalized as algebras with finite domains and a system is defined as a labelled transition system transforming environments. The pairs (query, decision) used as labels of the corresponding transition systems describe the interaction between the system and the policy that defines the decisions to be taken for a request in a given environment. We specified the conditions a policy should satisfy to be applied to a system and we showed how systems that are secure with respect to a policy can be defined.

The environments as well as the security policies can be defined in an abstract way or using a more concrete presentation. More precisely, rewrite systems are used in a standard way to perform function evaluation in the considered environments. Moreover, given a system and a compatible policy, constrained term rewriting systems are used to combine the necessary check for values in the current environment (performed through constraint solving) with the computation of a decision (performed through rewriting). This approach clarifies and improves previous works by a more appropriate treatment of the environment in which a security policy is applied.

The formalism we have proposed provides a unified representation of all objects related to the specification of a security policy and of the systems it can be applied on. We also propose a transformation of environments that allows the translation of an environment into another one and thus the possibility to study some properties which cannot be expressed only within the initial presentation. We have shown, in particular, how confidentiality, integrity and confinment can be expressed for the BLP policy that does not deal explicitly with information flows but only with objects containing tractable information. This provides some evidence that the proposed framework is also adequate to handle a large class of policies and to formalize a certain comparison between policies.

5.1 Related work

An important part of security is brought by access control whose aim is to prevent resources from unauthorized accesses. An access control policy is usually presented as a set of rules specifying that a given subject is allowed (or denied) to perform an action on a given object. Different access control models have been proposed to specify the authorization mechanism: discretionary [HRU76] and mandatory [Bib75, BL96b] access control models, role-based access control (RBAC) [SCFY96], organization-based access control (OrBAC) [KBB$^+$03], Chinese-Wall policy [BN89]... Such models are useful to analyze whether some security property (such as confidentiality or separation of duties) is satisfied in all possible states of the system.

Even in more general contexts than access control, policies are often expressed by rules in natural language: if some conditions are satisfied, then, grant or deny some request. Not surprisingly, their specification and verification can rely on various logic-based formalisms. The existing approaches have adopted different semantic frameworks such as first-order logic [HW03, JSSS01, HJM08, JM06], Datalog [BS02, LM03, BO05, DFK06], temporal logic [BBFS98, CCBS05], deontic logic [CC97, KBB$^+$03], or defeasible logic [MAG04]. In open and distributed property-based access control systems, tasks are delegated and agents must rely on others to perform actions that they themselves cannot do. The concept of trust has been formalized and models of trust together with logical formalisms have been proposed, such as for instance Wooldridge's Logic Of Rational Agents [JJ06]. These formalisms provide different degrees of expressiveness for authoring a policy.

In recent years, term rewriting theory has been applied for the specification and analysis of security policy in [DKKS07, BF08b, BF08a, SdO08, KKSdO08]. In a rewrite-based specification, policies are expressed by rules close to natural language: if some conditions are satisfied, then, a given request is evaluated into a decision; for instance, it may be granted or denied. Moreover strategic rewriting is used to express control on the rules and to handle priorities or choices between possible decisions. The rewrite-based approach provides executable specifications for security policies, which can be independently designed, verified, and then anchored on programs [dOWKK07] using a modular discipline. Once the policies are specified as rewriting systems, some properties can be checked, such as consistency of a policy that generally corresponds to the confluence of the underlying rewriting system while the completeness depends on the termination of the corresponding rewriting system. In [CMO08], the rewriting approach has been used to explore the information flow over the reachable states of a system and detect information leakage.

5.2 Further work

The proposed framework and especially the use of constrained term rewriting systems opens the way to further research problems related to the verification of properties of a security policy and of a secure system *w.r.t.* the policy . For instance, we expect to give some sufficient syntactic conditions, easy to check, to ensure totality and consistency of a security policy. We also need to further explore the power of environment transformation for proving properties over reachable states in the secure system. Moreover, we are currently working on an implementation of the framework.

A challenge in the domain of security policies is to dynamically integrate different security policies in a modular way. The policies and resources of a system may be modeled, built or owned by different unrelated parties, at different times, and under different environments. Hence, in order to avoid severe interference in their individual developments, it is better for a policy to adopt

a component-based architecture. To build a complex global policy, they are integrated via various composition operators.

A first goal is to define in our formalism a composition mechanism and to identify its general properties. A second objective is to study and to formalize several ways to compose policies and systems, to compare them and to express in a mathematical setting the properties that these composition mechanisms should satisfy. We also plan to define operators over policies allowing to solve conflicts or unspecified cases that may happen during composition and to classify these operators in order to characterize operators leading to a more restrictive policy or to a less restrictive policy, by considering comparison mechanism over policies and systems. The ultimate goal is to provide a formal definition of composition in order to specify and analyze the composition of security policies, and thus to give some theoretical foundations to policies composition.

References

[BBFS98] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. ACM Trans. Database Syst., 23(3):231–285, 1998.

[BF08a] C. Bertolissi and M. Fernández. An algebraic-functional framework for distributed access control. In International Conference on Risks and Security of Internet and Systems (CRISIS 2008), Tozeur, Tunisia. Proceedings IEEE Xplorer to appear., 2008.

[BF08b] C. Bertolissi and M. Fernández. A rewriting framework for the composition of access control policies. In Proceedings of PPDP'08, Valencia. ACM Press, 2008.

[Bib75] K. Biba. Integrity considerations for secure computer systems. Technical Report TR-3153, Mitre, Bedford, MA, 1975.

[BL73] D. Bell and L. LaPadula. Secure Computer Systems: a Mathematical Model. Technical Report MTR-2547 (Vol. II), MITRE Corp., Bedford, MA, May 1973.

[BL96a] D. Bell and L. LaPadula. Secure Computer Systems: a Mathematical Model. Journal of Computer Security, 4:239–263, 1996.

[BL96b] D.E. Bell and L.J. LaPadula. Secure computer systems: A mathematical model, volume ii. Journal of Computer Security, 4(2/3):229–263, 1996.

[BN89] D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In Proc. IEEE Symposium on Security and Privacy, pages 206–214, 1989.

[BO05] P.A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In Proceedings IEEE International Workshop on Policies for Distributed Systems and Networks, (POLICY). IEEE Society, 2005.

[BS02] P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. Journal of Computer Security, 10(3):241–272, 2002.

[CC97] Laurence Cholvy and Frédéric Cuppens. Analyzing consistency of security policies. In IEEE Symposium on Security and Privacy, pages 103–112, 1997.

[CCBS05] F. Cuppens, N. Cuppens-Boulahia, and T. Sans. A security model with non-atomic actions and deadlines. In CSFW, pages 186–196. IEEE Society, 2005.

[CMO08] Horatiu Cirstea, Pierre-Etienne Moreau, and Anderson Santana de Oliveira. Rewrite based specification of access control policies. In Daniel Dougherty and Santiago Escobar, editors, 3rd International Workshop on Security and Rewriting Techniques -

SecRet'08, volume 234 of Electronic Notes in Theoretical Computer Science, pages 37–54, Pittsburgh, PA, USA, June 2008. Elsevier.

[DFK06] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In Ulrich Furbach and Natarajan Shankar, editors, IJCAR, volume 4130 of Lecture Notes in Computer Science, pages 632–646. Springer, 2006.

[DKKS07] Daniel J. Dougherty, Claude Kirchner, Hélène Kirchner, and Anderson Santana de Oliveira. Modular access control via strategic rewriting. In Proceedings of 12th European Symposium On Research In Computer Security (ESORICS'07), Sep 2007.

[dOWKK07] Anderson Santana de Oliveira, Eric Ke Wang, Claude Kirchner, and Hélène Kirchner. Weaving rewrite-based access control policies. In FMSE '07: Proceedings of the 2007 ACM workshop on Formal methods in security engineering, pages 71–80, New York, NY, USA, 2007. ACM.

[EM85] .H Ehrig and B. Mahr. Fundamentals of Algebraic Specification I. Springer-Verlag New York, Inc., 1985.

[End72] H.B. Enderton. A mathematical introduction to logic. Academic Press, New York, 1972.

[HJM08] L. Habib, M. Jaume, and C. Morisset. A formal comparison of the bell & lapadula and rbac models. In Fourth International Symposium on Information Assurance and Security (IAS'08), pages 3–8. IEEE CS Press, 2008.

[HRU76] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. Commun. ACM, 19(8):461–471, 1976.

[HW03] Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. In CSFW, pages 187–201, 2003.

[JJ06] Bevan Jarvis and Lakhmi Jain. Trust in LORA: Towards a formal definition of trust in BDI agents. In Knowledge-Based Intelligent Information and Engineering Systems, volume 4252 of Lecture Notes in Computer Science, pages 458–463. Springer Berlin / Heidelberg, 2006.

[JM06] M. Jaume and C. Morisset. Towards a formal specification of access control. In P. Degano, R. Kusters, L. Vigano, and S. Zdancewic, editors, Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis FCS-ARSPA'06, pages 213–232, 2006.

[JSSS01] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Trans. Database Syst., 26(2):214–260, 2001.

[KBB+03] A. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In Proceedings IEEE 4th International Workshop on Policies for Distributed Systems and Networks, (POLICY), pages 120–131. IEEE Society, 2003.

[KKR90] Claude Kirchner, Hélène Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. Revue d'Intelligence Artificielle, 4(3):9–52, 1990.

[KKSdO08] C. Kirchner, H. Kirchner, and A. Santana de Oliveira. Analysis of rewrite-based access control policies. In D. Dougherty and S. Escobar, editors, Security and Rewriting Techniques, 3rd International Workshop Secret'08, volume informal proceedings, sub-

mitted to ENTCS, pages 37–51, 2008.

[LM03] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In PADL, pages 58–73, 2003.

[MAG04] Michael McDougall, Rajeev Alur, and Carl A. Gunter. A model-based approach to integrating security policies for embedded devices. In ACM EMSOFT, 2004.

[SCFY96] R. Shandu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. IEEE Computer, 29(2):38–47, 1996.

[SdO08] A. Santana de Oliveira. Réécriture et modularité pour les politiques de sécurité. PhD thesis, UHP Nancy 1, 2008.