Security rules *versus* Security properties

Mathieu Jaume

SPI – LIP6 – University Pierre & Marie Curie,
4 Place Jussieu, 75252 Paris Cedex 05, France
Mathieu.Jaume@Lip6.fr

Abstract. There exist many approaches to specify and to define security policies. We present here a framework in which the basic components of security policies can be expressed, and we identify their role in the description of a policy, of a system and of a secure system. In this setting, we formally describe two approaches to define policies, and we relate them: the rule-based approach consists of specifying the conditions under which an action is granted and, the property-based approach consists of specifying the security properties the policy aims to enforce. We also show how a policy can be applied to constrain an existing system, and how a secure system can be defined from a security policy.

Keywords: Security policies, security properties, security rules, systems.

1 Introduction

Security has become a major issue in computer science and there exists now a large collection of literature on this topic. In this context, many security policies have been introduced, describing, in a more or less formal way, within a particular specification language, a notion of information system, suitable in a particular context, and/or the specification of granted actions in this system, and/or the specification of secure states of this system. A classification of these approaches can be obtained by considering two main criteria. The first one is the language used to describe the policy. Specification languages can be natural languages, XACML, logic, reduction systems, automata, etc. Of course, a formal language, based on a clear syntax and semantics, allows a clear meaning of the described policy, and also allows to reason about its properties in a mathematical setting. Furthermore, depending on the language used to specify a policy, an operational mechanism, allowing to enforce the policy on a system, can be more or less difficult to develop. The second criterion is the definition of what is a security policy. Is it a specification of secure states of the system? is it an operational mechanism allowing to grant or to revoke actions on the system? does this mechanism take into account how the system is transformed by the actions, or is it just based on the name of actions? what is a security information? is it an information that the policy aims to control or is it an information that the policy uses to control a system? what is a security configuration? can

a security configuration be modified during the lifetime of a system? is there a (administrative) policy to control changes of security configurations? does a policy take into account a notion of environment? are the systems on which policies apply and the policies independent, or do they share some entities? In this paper, we provide a formal framework allowing to specify and to define security policies by following several approaches and we show how to relate them. In this setting, we focus on two approaches to define policies: the rule-based approach consists of specifying the conditions under which an action is granted, and the property-based approach consists of specifying the security properties the policy aims to enforce. As we will see, while property-based policies can be expressed as rule-based policies, there exist some rule-based policies which cannot be expressed as property-based policies. Indeed rule-based policies allow to specify some behaviours which cannot be enforced by property-based policies. This is the main difference between the two approaches. However, we give here a formal characterization of the class of rule-based policies which can be expressed as property-based policies. Moreover, we also present how to apply a policy to constrain an existing system to get a secure system according to the policy, and how to define a secure system from a policy. Last, we give some equivalence results about the executions of such systems. Our framework does not correspond to the definition of a language to deal with security policies but aims to define what are the components needed to specify security policies, what are their roles and, depending on the considered approach, what can be done with a security policy. Using such a framework provides several benefits. Firstly, policies developed within this framework can be easily reused when considering new policies or even variant of these policies. Secondly, we think that this framework provides some methodological guidelines allowing to specify security policies. Finally, it is convenient to deal with a generic formal framework in which many policies can be expressed in order to perform analysis of these policies and to define operations over these policies (such as comparison or composition of policies).

2 Systems

This section introduces the basic notions related to labelled transition systems (LTS) together with their notations. A LTS $\mathbb{S}$ is a tuple $(\Sigma, \Sigma^0, L, \delta)$ where Σ is a set of states, $\Sigma^0 \subseteq \Sigma$ is a set of initial states, L is a set of labels, and $\delta \subseteq \Sigma \times L \times \Sigma$ is a transition relation. Often, we will write $\sigma_1 \xrightarrow{l}_\delta \sigma_2$ instead of $(\sigma_1, l, \sigma_2) \in \delta$. Furthermore, we will write $\overrightarrow{l}$ a sequence $(l_1, \cdots, l_n)$ of labels in L and, when it is defined, we will write $\sigma_1 \xrightarrow{\overrightarrow{l}}{}^\star_\delta \sigma_{n+1}$ the sequence:

$$\sigma_1 \xrightarrow{l_1}_\delta \sigma_2 \xrightarrow{l_2}_\delta \cdots \xrightarrow{l_{n-1}}_\delta \sigma_n \xrightarrow{l_n}_\delta \sigma_{n+1}$$

of transitions. From an operational point of view, we can require that the relation δ of a LTS defines a deterministic and left-total relation:

$$\forall \sigma, \sigma_1, \sigma_2 \in \Sigma\ \forall l \in L\ (\sigma \xrightarrow{l}_\delta \sigma_1 \wedge \sigma \xrightarrow{l}_\delta \sigma_2) \Rightarrow \sigma_1 = \sigma_2$$
$$\forall \sigma_1 \in \Sigma\ \forall l \in L\ \exists \sigma_2 \in \Sigma\ \sigma_1 \xrightarrow{l}_\delta \sigma_2$$

We write $\Gamma(\mathbb{S})$ the set of states that are reachable by "applying" a finite number of times δ from a state in Σ^0. We also define the set $Exec(\mathbb{S}) \subseteq \Sigma^+$ (where Σ^+ is the set of non-empty finite sequences of states) of executions of $\mathbb{S}$ as follows:

$$Exec(\mathbb{S}) = \{(\sigma_1, \cdots, \sigma_n) \mid n \geq 1 \wedge \sigma_1 \in \Sigma^0 \wedge \forall i \ \exists l \in L \ \sigma_i \xrightarrow{l}_\delta \sigma_{i+1}\}$$

Example 1. We define the system $\mathbb{S}_{ac} = (\Sigma_{ac}, \Sigma^0_{ac}, L_{ac}, \delta_{ac})$ allowing to add or to release accesses done by active entities, the subjects in $\mathcal{S}$, over passive entities, the objects in $\mathcal{O}$, according to access modes in $\mathcal{A}$ (for example, read, write, etc). A state is represented by a function $\alpha : \mathcal{S} \to \wp(\mathcal{O} \times \mathcal{A})$ such that $(o, a) \in \alpha(s)$ means that s has an access over o according to the access mode a. Hence, $\Sigma_{ac} = \{\alpha : \mathcal{S} \to \wp(\mathcal{O} \times \mathcal{A})\}$. We define $\Sigma^0_{ac} = \{\alpha_0 \mid \forall s \in \mathcal{S} \ \alpha_0(s) = \emptyset\}$ expressing that no access is done in the initial state. By considering the set of labels $L_{ac} = \{\langle +, s, o, a\rangle, \langle -, s, o, a\rangle \mid s \in \mathcal{S}, \ o \in \mathcal{O}, \ a \in \mathcal{A}\}$, where $\langle +, s, o, a\rangle$ (resp. $\langle -, s, o, a\rangle$) allows to add (resp. to release) the access done by s over o according to a, we define the transition relation δ_{ac} by:

$$\left\{ \begin{array}{l} \alpha \xrightarrow{\langle +,s,o,a\rangle}_{\delta_{ac}} \alpha[s \leftarrow \alpha(s) \cup \{(s, o, a)\}] \\ \alpha \xrightarrow{\langle -,s,o,a\rangle}_{\delta_{ac}} \alpha[s \leftarrow \alpha(s) \backslash \{(s, o, a)\}] \end{array} \right\} \text{where } f[x \leftarrow v](y) = \left\{ \begin{array}{ll} f(y) & \text{if } x \neq y \\ v & \text{if } x = y \end{array} \right.$$

We introduce now the notion of interpretation of the states of a system: an interpretation $I^{\mathbb{D}}_\Sigma$ of Σ based on the domain $\mathbb{D}$ is a mapping from Σ to $\mathbb{D}$. We write $[\![\sigma]\!]_{I^{\mathbb{D}}_\Sigma}$ the interpretation of σ, and we extend this definition for subsets Σ' of Σ as follows: $[\![\Sigma']\!]_{I^{\mathbb{D}}_\Sigma} = \{[\![\sigma]\!]_{I^{\mathbb{D}}_\Sigma} \mid \sigma \in \Sigma'\}$. The interpreted system is defined by $[\![\mathbb{S}]\!]_{I^{\mathbb{D}}_\Sigma} = ([\![\Sigma]\!]_{I^{\mathbb{D}}_\Sigma}, [\![\Sigma^0]\!]_{I^{\mathbb{D}}_\Sigma}, L, \delta^{I^{\mathbb{D}}_\Sigma})$ where $\delta^{I^{\mathbb{D}}_\Sigma} = \{([\![\sigma_1]\!]_{I^{\mathbb{D}}_\Sigma}, l, [\![\sigma_2]\!]_{I^{\mathbb{D}}_\Sigma}) \mid \sigma_1 \xrightarrow{l}_\delta \sigma_2\}$.

Example 2. We can change the representation of states of $\mathbb{S}_{ac}$ by considering the interpretation $I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}$ of Σ_{ac} where $\mathbb{A}_{ac} = \wp(\mathcal{S} \times \mathcal{O} \times \mathcal{A})$ and:

$$\forall \alpha \in \Sigma_{ac} \ [\![\alpha]\!]_{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}} = \bigcup_{s \in \mathcal{S}} \bigcup_{(o,a) \in \alpha(s)} \{(s, o, a)\}$$

In this way, states of the interpreted system are sets of accesses.

Intuitively, interpretations will be useful when applying security policies over systems: an interpretation can provide the information that a policy aims to control. Indeed, thanks to interpretations, it will be possible to apply a policy on a system even if the policy and the system do not share the same entities or representations. For example, by considering an interpretation whose mapping provides the information flows generated by a set of accesses, it is possible to apply a flow policy over an access system such as $\mathbb{S}_{ac}$. Furthermore, considering a composition operator over the domain of an interpretation allows to define a semantics over executions of a system. This can be useful to check that even if each transition of a sequence is secure according to a policy, the sequence is also secure according to the policy. Such property does not always hold. For example, when dealing with a flow policy over an access system, some sequences

of legal sets of accesses (e.g. sets of accesses generating legal flows according to the policy) may generate, by composition of flows generated by each element of the sequence, some illegal flows. Later, when defining sytems from policies, a transition will be labelled both by a request submitted to the system and by the answer given by the policy to this request. We write $\mathcal{R}$ for the set of requests, and $\mathcal{D}$ for the set of possible answers. The semantics of $\mathcal{R}$ and $\mathcal{D}$ is defined by a relation $[\![\mathcal{R}]\!]_{\Sigma}^{\mathcal{D}} \subseteq \Sigma \times (\mathcal{R} \times \mathcal{D}) \times \Sigma$. $(\sigma_1, (R, d), \sigma_2) \in [\![\mathcal{R}]\!]_{\Sigma}^{\mathcal{D}}$ means that when the system is in the state σ_1, if the request R is applied according to the answer d, then the system moves into the state σ_2. The notion of answers allows to consider policies specifying transformations that must be done over states when a request has not been accepted (this can be useful for policies that aim to control the number of times that an entity tries to perform an action that would lead to an insecure state). However, for many policies, the set of answers contains only two elements allowing to specify that the request is granted or not.

Example 3. We define the set $\mathcal{R}^{ac}$ as the set L_{ac} and the set $\mathcal{D} = \{\text{yes}, \text{no}\}$. The semantics of $\mathcal{R}^{ac}$ can be defined by:

$$(A_1, (\langle +, s, o, a\rangle, \text{yes}), A_2) \in [\![\mathcal{R}^{ac}]\!]_{\mathbb{A}_{ac}}^{\mathcal{D}} \Leftrightarrow A_2 = A_1 \cup \{(s, o, a)\}$$
$$(A_1, (\langle -, s, o, a\rangle, \text{yes}), A_2) \in [\![\mathcal{R}^{ac}]\!]_{\mathbb{A}_{ac}}^{\mathcal{D}} \Leftrightarrow A_2 = A_1 \backslash \{(s, o, a)\}$$
$$(A_1, (R, \text{no}), A_2) \in [\![\mathcal{R}^{ac}]\!]_{\mathbb{A}_{ac}}^{\mathcal{D}} \Leftrightarrow A_1 = A_2$$

3 Security policies

Several points of view exist on security policies, among which two main approaches can be distinguished.

Property-based security policies. A property-based policy is a characterization of secure elements of a set according to some security information. Hence, specifying a property-based policy $\mathbb{P}$ first consists of defining a set $\mathbb{A}$ of "things" that the policy aims to control, called the security targets, in order to ensure the desired security properties (these "things" can be the actions simultaneously done in the system or some information about the entities of the system). Then, a set $\mathcal{C}$ of security configurations is introduced: configurations correspond to the information needed to characterize secure elements of $\mathbb{A}$ according to the policy. Last, a policy is defined by a relation $\Vdash$ between configurations and targets allowing to express that, given a configuration, a target satisfies the policy.

Definition 1. *A property-based security policy $\mathbb{P}$ is a tuple $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \Vdash)$ where $\mathbb{A}$ is a set of security targets, $\mathcal{C}$ is a set of security configurations and $\Vdash \subseteq \mathcal{C} \times \mathbb{A}$ is a relation specifying secure targets according to configurations.*

Example 4. We consider here the HRU policy $\mathbb{P}_{\mathsf{hru}} = (\mathbb{A}_{ac}, \mathcal{C}_{\mathsf{hru}}, \Vdash_{\mathbb{P}_{\mathsf{hru}}})$, introduced in [14], which is a discretionary access control policy, aiming to control accesses done in a system. The set of security targets is $\mathbb{A}_{ac}$ (defined in example 2) and the set of security configurations is $\mathcal{C}_{\mathsf{hru}} = \mathbb{A}_{ac}$ (a configuration $m_D \in \mathcal{C}_{\mathsf{hru}}$ specifies a set of granted accesses). Now we can define secure targets as sets of accesses which are granted: $m_D \Vdash_{\mathbb{P}_{\mathsf{hru}}} A$ iff $A \subseteq m_D$.

In fact, within the property-based approach, a policy $\mathbb{P} = (\mathbb{A}_{\mathbb{P}}, \mathcal{C}_{\mathbb{P}}, \Vdash_{\mathbb{P}})$ can be viewed as the definition of a semantics for $\mathcal{C}$: each configuration c denotes the set of targets $[\![c]\!]_{\mathbb{P}} = \{A \in \mathbb{A}_{\mathbb{P}} \mid c \Vdash_{\mathbb{P}} A\}$ that c authorizes. Such definition is similar to the one introduced in [3], in the context of access control. For example, if we consider the HRU policy, we have $[\![m_D]\!]_{\mathbb{P}_{\mathsf{hru}}} = \wp(m_D)$. Note that it may be useful to consider systems allowing to modify configurations. For these systems, we can also introduce a policy allowing to control the transformations of configurations. Hence, the "things" that this policy aims to control is defined by the set $\mathcal{C}_{\mathbb{P}}$. Such a policy is often called an administrative policy for $\mathbb{P}$ and is defined as a policy $\mathbb{P}_{\mathbb{P}} = (\mathcal{C}_{\mathbb{P}}, \mathcal{C}, \Vdash_{\mathbb{P}_{\mathbb{P}}})$. In the following, given a set $\mathcal{R}$ of requests together with its semantics $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$, a policy $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \Vdash)$, and a configuration $c \in \mathcal{C}$, we write:

$$A_0 \xrightarrow{((R_1,d_1),\cdots,(R_k,d_k))}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathbb{P},c} A_k$$

to express that there exist $A_1, \cdots, A_{k-1} \in \mathbb{A}$ such that:

$$A_0 \xrightarrow{(R_1,d_1)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A_1 \xrightarrow{(R_2,d_2)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} \cdots \xrightarrow{(R_k,d_k)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A_k \text{ and } \forall i\ (0 \leq i \leq k)\ \ c \Vdash A_i$$

Rule-based security policies. A security policy can also be viewed as a description of the conditions under which an action is permitted or forbidden. We extend here such an approach by considering an arbitrary set of answers. As before, a policy is defined from a set $\mathbb{A}$ of security targets and a set $\mathcal{C}$ of security configurations. Furthermore, to define a policy by specifying a set of authorized actions, we introduce a set $\mathcal{R}$ of requests (corresponding to names of actions) and a set $\mathcal{D}$ of answers (corresponding to different authorizations). Now, the policy is defined by a relation $\Vdash\!\!\!\vdash \subseteq (\mathcal{C} \times \mathbb{A}) \times (\mathcal{R} \times \mathcal{D})$, where $(c, A) \Vdash\!\!\!\vdash (R, d)$ means that when the current configuration is c and the current target is A, the action R can be performed according to the answer d.

Definition 2. *A rule-based security policy $\mathfrak{P}$ is a tuple $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash\!\!\!\vdash)$ where $\mathbb{A}$ is the set of security targets, $\mathcal{C}$ is the set of security configurations, $\mathcal{R}$ is the set of requests, $\mathcal{D}$ is a set of answers and $\Vdash\!\!\!\vdash \subseteq (\mathcal{C} \times \mathbb{A}) \times (\mathcal{R} \times \mathcal{D})$ is a relation specifying which (and how) actions can be performed.*

Example 5. The HRU policy can be defined as the rule-based policy $\mathfrak{P}_{\mathsf{hru}} = (\mathbb{A}_{ac}, \mathcal{C}_{\mathsf{hru}}, \mathcal{R}^{ac}, \mathcal{D}, \Vdash\!\!\!\vdash_{\mathfrak{P}_{\mathsf{hru}}})$ where $\mathcal{D} = \{\text{yes}, \text{no}\}$ and $(m_D, A) \Vdash\!\!\!\vdash_{\mathfrak{P}_{\mathsf{hru}}} (R, d)$ iff:

$$\begin{array}{l}(R = \langle -, s, o, a\rangle \wedge d = \text{yes}) \vee (R = \langle +, s, o, a\rangle \wedge d = \text{yes} \wedge (s, o, a) \in m_D)\\ \vee (R = \langle +, s, o, a\rangle \wedge d = \text{no} \wedge (s, o, a) \notin m_D)\end{array}$$

Hence, when $\mathcal{D} = \{\mathsf{yes}, \mathsf{no}\}$, given a pair (c, A), the set of granted (resp. forbidden) actions is the set $\{R \in \mathcal{R} \mid (c, A) \Vdash\!\!\!\vdash (R, \mathsf{yes})\}$ (resp. $\{R \in \mathcal{R} \mid (c, A) \Vdash\!\!\!\vdash (R, \mathsf{no})\}$). Such definition is similar to the one introduced in [6], where a policy is defined in terms of authorized actions, and can be extended by considering the set $[\![c, A]\!]_{\mathfrak{P}} = \{(R, d) \in \mathcal{R} \times \mathcal{D} \mid (c, A) \Vdash\!\!\!\vdash (R, d)\}$. Here again, administrative policies can be defined for rule-based policies. More generally, an administrative policy for a (rule-based or property-based) security policy whose set of security

configurations is $\mathcal{C}$ can be a (rule-based or property-based) policy whose set of security targets is $\mathcal{C}$. Given a policy $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$, a security configuration $c \in \mathcal{C}$, and a relation $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ specifying the semantics of requests, we write:

$$A_0 \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A_1 \qquad \left(\text{resp. } A_0 \xrightarrow{((R_1,d_1),\cdots,(R_k,d_k))}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A_k\right)$$

to express that:

$$A_0 \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A_1 \text{ and } (c, A_0) \Vdash (R, d)$$
$$\left(\begin{array}{l} \exists A_1, \cdots, A_{k-1} \in \mathbb{A} \\ \text{resp. } \left(\begin{array}{l} A_0 \xrightarrow{(R_1,d_1)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A_1 \xrightarrow{(R_2,d_2)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} \cdots \xrightarrow{(R_k,d_k)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A_k \\ \wedge\ \forall i\ (0 \leq i \leq k-1)\ \ (c, A_i) \Vdash (R_{i+1}, d_{i+1}) \end{array}\right) \end{array}\right)$$

Within the rule-based approach, when $\mathcal{D} = \{\mathsf{yes}, \mathsf{no}\}$, two main issues must be handled. Given a current configuration and target, for each request, the policy must provide one answer (is the action permitted or forbidden?) and only one answer (no action can be both permitted and forbidden):

- $\mathfrak{P}$ is complete iff: $\forall R \in \mathcal{R}\ \exists d \in \mathcal{D} \quad (c, A) \Vdash (R, d)$
- $\mathfrak{P}$ is consistent iff: $((c, A) \Vdash (R, d_1) \wedge (c, A) \Vdash (R, d_2)) \Rightarrow d_1 = d_2$

For example, $\mathfrak{P}_{\mathsf{hru}}$ is both complete and consistent. When considering an arbitrary set of answers, only the first issue must be handled: the policy must provide an answer for each request, but several answers can be given to this request, all of them corresponding to authorized ways of applying the request (in practice, a mechanism allowing to specify which answer must be considered when several answers are given by the policy can be introduced, for example, an order relation over $\mathcal{D}$). The relation $\Vdash$ characterizes "secure" pairs (R, d) by considering a pair (c, A) describing the current state of a system. For a class of policies, $\Vdash$ can be defined only from the configuration c. For example, this is the case for the HRU policy for which deciding if a request is granted or not can be done by only considering the set m_D of authorized accesses. We call such policies free policies. Note that there exist non-free policies, such as MLS (MultiLevel Security) access control policies for which deciding if adding an access is granted or not is done by considering both the configuration (to ensure that the security level of the subject authorizes the access over the object according to its security level) and the current security target (to ensure that the new access won't generate an illegal information flow between objects, according to security levels of objects). Formally, a rule-based policy $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$ is said to be free iff:

$$\forall c \in \mathcal{C}\ \forall R \in \mathcal{R}\ \forall d \in \mathcal{D}\ \forall A_1, A_2 \in \mathbb{A} \quad (c, A_1) \Vdash (R, d) \Leftrightarrow (c, A_2) \Vdash (R, d)$$

Note that some frameworks, like in [6], only allow to consider free policies. We introduce now a property, based on the semantics of requests. Intuitively, this property holds iff for all reachable targets A_1 and A_2, according to the policy, if the semantics of $\mathcal{R}$ contains a transition allowing to transform A_1 into A_2, then

this transformation is granted by the policy. More formally, given a relation $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ specifying the semantics of $\mathcal{R}$, and a set $\mathscr{I} \subseteq \mathcal{C} \times \mathbb{A}$ containing pairs (c, A) such that A is assumed to be secure according to c, $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$ satisfies the switching property according to $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ and $\mathscr{I}$ iff:

$$\forall (c, A_0), (c, A_0') \in \mathscr{I}\ \forall A, A' \in \mathbb{A}\ \forall (R, d) \in \mathcal{R} \times \mathcal{D}\ \forall \overrightarrow{(R_1, d_1)}, \overrightarrow{(R_2, d_2)} \in (\mathcal{R} \times \mathcal{D})^\star$$
$$\left(A_0 \xrightarrow{\overrightarrow{(R_1,d_1)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A \wedge A_0' \xrightarrow{\overrightarrow{(R_2,d_2)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A' \wedge A \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A' \right)$$
$$\Rightarrow (c, A) \Vdash (R, d)$$

For example, the policy $\mathfrak{P}_{\mathsf{hru}}$ satisfies the switching property. Of course, there exist some policies for which the switching property does not hold. This is the case for some policies aiming to control the order of actions.

Property-based policies *versus* **Rule-based policies.** We relate here the two approaches introduced above by showing how to obtain a property-based policy from a rule-based policy and *vice-versa* and we prove some equivalence results.

Building a property-based policy from a rule-based policy leads to characterize secure elements of $\mathbb{A}$ from granted actions according to a security configuration. To achieve this goal, both the semantics of requests and a set $\mathscr{I}$ of "initial" pairs of the form (c, A) considered as secure must be provided. In this way, a secure target A according to a configuration c is a reachable target by applying $\rightarrow_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c}$ from an element A_0 such that $(c, A_0) \in \mathscr{I}$. More formally, let $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$ be a rule-based policy, $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ be a relation specifying the semantics of $\mathcal{R}$, and $\mathscr{I} \subseteq \mathcal{C} \times \mathbb{A}$ be a set of pairs (c, A). We define the $(\mathfrak{P}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathscr{I})$-policy as the property-based policy $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \vdash)$ where:

$$c \vdash A \Leftrightarrow \left(\exists (c, A_0) \in \mathscr{I}\ \exists \overrightarrow{(R, d)} \in (\mathcal{R} \times \mathcal{D})^\star\ A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A \right)$$

For example, the policy $\mathbb{P}_{\mathsf{hru}}$ is the $(\mathfrak{P}_{\mathsf{hru}}, [\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}}, \mathscr{I})$-policy where $\mathscr{I} = \{(c, \emptyset) \mid c \in \mathcal{C}_{\mathsf{hru}}\}$. We prove now that, given a configuration, applying a granted action according to an answer over a secure target leads to a secure target, and conversely, given a configuration c, if the semantics of the language of requests contains a transition allowing to transform a secure target A_1 into a secure target A_2, then such a transformation is granted by the policy (in order to prove this property, we need to suppose that the rule-based policy satisfies the switching property).

Proposition 1. *Let $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \vdash)$ be the $(\mathfrak{P}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathscr{I})$-policy, $c \in \mathcal{C}$, $A_0, A \in \mathbb{A}$ and $\overrightarrow{(R, d)} \in (\mathcal{R} \times \mathcal{D})^\star$.*

1. *If $(c, A_0) \in \mathscr{I}$ and $A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A$, then $A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathbb{P},c} A$.*
2. *If $\mathfrak{P}$ satisfies the switching property according to $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ and $\mathscr{I}$, then:*

$$A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathbb{P},c} A \Rightarrow A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} A$$

Hence, a policy $\mathfrak{P}$, which does not satisfy the switching property, can be more restrictive than $\mathbb{P}$ and cannot be expressed as an "equivalent" property-based policy. However, by adding some information about "the past" into targets (for example the sequence of transformations that have been done from an initial target to obtain the current target), it becomes possible to solve this problem.

Building a rule-based policy from a property-based policy $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \Vdash)$ leads to introduce a set $\mathcal{R}$ of requests together with its semantics $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$. We define the $(\mathbb{P}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}})$-policy as the rule-based policy $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash\!\!\!\vdash)$ where:

$$(c, A) \Vdash\!\!\!\vdash (R, d) \Leftrightarrow (\exists A' \in \mathbb{A}\ A \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} A' \wedge c \Vdash A')$$

For example, the $(\mathbb{P}_{\mathsf{hru}}, [\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}})$-policy is based on a relation $\Vdash\!\!\!\vdash$ defined by:

$$(m_D, A) \Vdash\!\!\!\vdash (R, d) \Leftrightarrow \begin{pmatrix} (R = \langle +, s, o, a\rangle \wedge d = \mathsf{yes} \wedge A \subseteq m_D \wedge (s, o, a) \in m_D) \\ \vee\ (R = \langle +, s, o, a\rangle \wedge d = \mathsf{no} \wedge (A \not\subseteq m_D \vee (s, o, a) \notin m_D)) \\ \vee\ (R = \langle -, s, o, a\rangle \wedge d = \mathsf{yes} \wedge A \backslash \{(s, o, a)\} \subseteq m_D) \\ \vee\ (R = \langle -, s, o, a\rangle \wedge d = \mathsf{no} \wedge A \backslash \{(s, o, a)\} \not\subseteq m_D) \end{pmatrix}$$

This relation is slightly different from the relation $\Vdash\!\!\!\vdash_{\mathfrak{P}_{\mathsf{hru}}}$ of the policy $\mathfrak{P}_{\mathsf{hru}}$ (example 5). However, the only difference is concerned with answers for requests when the current target is not secure according to the configuration. Hence, as we will see, when applied over a system whose initial states are secure, these two policies lead to the same secure states. Last, we state the following proposition.

Proposition 2. *Let* $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash\!\!\!\vdash)$ *be the* $(\mathbb{P}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}})$*-policy,* $A_0, A \in \mathbb{A}$*,* $c \in \mathcal{C}$ *and* $\overrightarrow{(R, d)} \in (\mathcal{R} \times \mathcal{D})^\star$*.*

1. $A_0 \xrightarrow{\overrightarrow{(R,d)}}^\star_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathbb{P}, c} A \Rightarrow A_0 \xrightarrow{\overrightarrow{(R,d)}}^\star_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathfrak{P}, c} A$
2. *If* $c \Vdash A_0$ *and if* $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ *is deterministic, then*

$$A_0 \xrightarrow{\overrightarrow{(R,d)}}^\star_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathfrak{P}, c} A \Rightarrow A_0 \xrightarrow{\overrightarrow{(R,d)}}^\star_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathbb{P}, c} A$$

Note that, when $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ is deterministic, rule-based policies obtained from property-based policies satisfy the switching property.

4 Security policies and systems

We describe here how a policy can be applied on an existing system in order to obtain a secure system, and how to obtain a secure system from a policy.

Property-based policies and systems. To apply a property-based policy on a system, we first have to define an interpretation of the states of the system, providing information on which the policy can apply (states of the system are

interpreted as security targets of the policy). Then, applying a policy on a system leads to a system whose states are enriched with security configurations, whose initial states are secure initial states, and whose transition relation is obtained by removing all "illegal" transitions (e.g. transitions transforming a secure state into a non-secure state). During executions of this new system, security configurations are constant (labels are not concerned with configurations). Systems for which the configurations can evolve can be obtained by composition with a system on which an administrative policy can apply. In practice, applying a policy over a system may be done by considering a particular security configuration, or a particular set of configurations, to constrain the executions of the system.

Definition 3. *Let $\mathbb{S} = (\Sigma, \Sigma^0, L, \delta)$ be a system, $\mathbb{P} = (\mathbb{A}, \mathcal{C}, \Vdash)$ be a property-based security policy, C be a subset of $\mathcal{C}$, and $I^{\mathbb{A}}_{\Sigma}$ be an interpretation of Σ. We define the system $\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} = (C \times \Sigma, \lfloor \Sigma^0 \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C}, L, \lfloor \delta \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C})$ where:*

$$\lfloor \Sigma^0 \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} = \{(c,\sigma) \mid \sigma \in \Sigma^0 \wedge c \in C \wedge c \Vdash [\![\sigma]\!]_{I^{\mathbb{A}}_{\Sigma}}\}$$
$$\lfloor \delta \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} = \{((c,\sigma_1), l, (c,\sigma_2)) \mid \sigma_1 \xrightarrow{l}_{\delta} \sigma_2 \wedge c \in C \wedge (c \Vdash [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \Rightarrow c \Vdash [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}})\}$$

The use of an interpretation, which can be viewed as a kind of interface between the policy and the system, allows to apply a policy on several different systems.

Example 6. By considering $C = \mathcal{C}_{\mathsf{hru}}$ and the interpretation $I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}$ (introduced in example 2), applying the policy $\mathbb{P}_{\mathsf{hru}}$ on the system $\mathbb{S}_{ac}$ leads to a system whose states belongs to $\mathcal{C}_{\mathsf{hru}} \times \Sigma_{ac}$, whose labels are labels in L_{ac}, and defined by:

$$\lfloor \Sigma^0_{ac} \rfloor^{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}}_{\mathbb{P}_{\mathsf{hru}},\mathcal{C}_{\mathsf{hru}}} = \bigcup_{m \in \mathcal{C}_{\mathsf{hru}}} \{(m,\alpha) \mid \alpha \in \Sigma^0_{ac}\}$$

$$\lfloor \delta_{ac} \rfloor^{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}}_{\mathbb{P}_{\mathsf{hru}},\mathcal{C}_{\mathsf{hru}}}$$
$$= \left\{ \begin{array}{l} ((m,\alpha), \langle +, s, o, a\rangle, (m, \alpha[s \leftarrow \alpha(s) \cup \{(o,a)\}])) \mid \\ (\exists s' \in \mathcal{S}\ \exists o' \in \mathcal{O}\ \exists a' \in \mathcal{A}\ (o',a') \in \alpha(s') \wedge (s',o',a') \notin m) \vee (s,o,a) \in m \end{array} \right\}$$
$$\cup \{((m,\alpha), \langle -, s, o, a\rangle, (m, \alpha[s \leftarrow \alpha(s) \backslash \{(s,o,a)\}]))\}$$

Such a construction allows to ensure that all reachable states of $\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C}$ are secure with respect to the policy $\mathbb{P}$.

Proposition 3. $\forall (c,\sigma) \in \Gamma(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C})\ c \Vdash [\![\sigma]\!]_{I^{\mathbb{A}}_{\Sigma}}$

Defining a system from a property-based policy $\mathbb{P}$ can be done by considering a language of requests $\mathcal{R}$ together with its semantics $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$, and a subset $\mathbb{A}^0$ of $\mathbb{A}$, and by applying $\mathbb{P}$ (according to the interpretation $I^{\mathbb{A}}_{\mathbb{A}}$ of $\mathbb{A}$ such that $[\![A]\!]_{I^{\mathbb{A}}_{\mathbb{A}}} = A$) on the system $(\mathbb{A}, \mathbb{A}^0, \mathcal{R} \times \mathcal{D}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}})$. For example, from the policy $\mathbb{P}_{\mathsf{hru}}$, the set $\mathbb{A}^0_{ac} = \{\emptyset\} \subseteq \mathbb{A}_{ac}$, and the set $\mathcal{R}^{ac}$ of requests together with its semantics $[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}}$, we can define the system obtained in example 6 by applying $\mathbb{P}_{\mathsf{hru}}$ on $\mathbb{S}_{ac}$. However, note that such a construction leads to consider all the correct transitions according to $\mathbb{P}$ and to $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$, and, in some situations, it can

be useful to constrain the transition relation δ. For example, it can happen that a correct transition according to $\mathbb{P}$ does not belong to δ in order to avoid that some subjects perform some authorized actions. This is the case for some administrative policies where access rights over an object must be provided only by its owner. In this case, it suffices to consider a transition relation δ such that:

$$\delta \subseteq \left\lfloor [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}} \right\rfloor^{I^{\mathbb{A}}_{\mathbb{A}}}_{\mathbb{P},\mathcal{C}}$$

Rule-based policies and systems. To apply a rule-based policy $\mathfrak{P}$ on a system $\mathbb{S}$, we have to define an interpretation $I^{\mathbb{A}}_{\Sigma}$ of the states of the system whose domain is the set of security targets of $\mathfrak{P}$. In addition, to characterize initial states of the system, the definition of a subset of $\mathcal{C} \times \mathbb{A}$ must be considered. Last, since $\mathfrak{P}$ is defined in terms of granted actions, the semantics of the language of requests of $\mathfrak{P}$ must be given. We introduce here two ways to apply a rule-based policy on a system. The first one consists in removing all transitions $(\sigma_1, l, \sigma_2) \in \delta$ such that $[\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \neq [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}}$ and such that each transition labelled by a request allowing to transform $[\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}}$ into $[\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}}$ is not granted by $\mathfrak{P}$. The second one is similar but consists in considering "weak-simulations" of transitions of δ (e.g. a sequence of requests may be used to transform $[\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}}$ into $[\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}}$).

Definition 4. *Let $\mathbb{S} = (\Sigma, \Sigma^0, L, \delta)$ be a system, $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$ be a rule-based policy, $C \subseteq \mathcal{C}$, $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ be a relation specifying the semantics of $\mathcal{R}$, $I^{\mathbb{A}}_{\Sigma}$ be an interpretation of Σ, and $\mathscr{I}$ be a subset of $C \times \mathbb{A}$. We define the systems:*

$$\begin{aligned}
\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} &= \left(C \times \Sigma, \lfloor \Sigma^0 \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}, L, \lfloor \delta \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right) \\
\lfloor \mathbb{S}, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} &= \left((C \times \Sigma, \lfloor \Sigma^0 \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}, L, \lfloor \delta, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right)
\end{aligned}$$

where:

$$\begin{aligned}
&\lfloor \Sigma^0 \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} = \{\, (c,\sigma) \mid \sigma \in \Sigma^0 \wedge (c, [\![\sigma]\!]_{I^{\mathbb{A}}_{\Sigma}}) \in \mathscr{I} \,\} \\
&\lfloor \delta \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \\
&= \left\{ \begin{array}{l} ((c,\sigma_1), l, (c,\sigma_2)) \mid (\sigma_1, l, \sigma_2) \in \delta \wedge c \in C \\ \wedge\, [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \neq [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \Rightarrow \exists (R,d) \in \mathcal{R} \times \mathcal{D} \;\; [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \end{array} \right\} \\
&\lfloor \delta, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} = \left\{ \begin{array}{l} ((c,\sigma_1), l, (c,\sigma_2)) \mid (\sigma_1, l, \sigma_2) \in \delta \wedge c \in C \\ \wedge \exists \overrightarrow{(R,d)} \in (\mathcal{R} \times \mathcal{D})^\star \;\; [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c} [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \end{array} \right\}
\end{aligned}$$

Example 7. By considering $[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}}$, $\mathscr{I} = \mathcal{C}_{\mathsf{hru}} \times \{\emptyset\}$ and $I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}$, applying $\mathfrak{P}_{\mathsf{hru}}$ on $\mathbb{S}_{ac}$ leads to a system whose states belongs to $\mathcal{C}_{\mathsf{hru}} \times \Sigma_{ac}$ and defined by:

$$\begin{aligned}
&\lfloor \Sigma^0_{ac} \rfloor^{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}}_{\mathfrak{P}_{\mathsf{hru}},\mathcal{C}_{\mathsf{hru}},[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}},\mathscr{I}} = \bigcup_{m \in \mathcal{C}_{\mathsf{hru}}} \left\{ (m,\alpha) \mid \alpha \in \Sigma^0_{ac} \right\} \\
&\lfloor \delta_{ac} \rfloor^{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}}_{\mathfrak{P}_{\mathsf{hru}},\mathcal{C}_{\mathsf{hru}},[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}},\mathscr{I}} = \lfloor \delta_{ac}, \star \rfloor^{I^{\mathbb{A}_{ac}}_{\Sigma_{ac}}}_{\mathfrak{P}_{\mathsf{hru}},\mathcal{C}_{\mathsf{hru}},[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}},\mathscr{I}} = \\
&\quad \{((m,\alpha), \langle +, s, o, a\rangle, (m, \alpha[s \leftarrow \alpha(s) \cup \{(s,o,a)\}])) \mid (s,o,a) \in m\} \\
&\cup \{((m,\alpha), \langle -, s, o, a\rangle, (m, \alpha[s \leftarrow \alpha(s) \backslash \{(s,o,a)\}]))\}
\end{aligned}$$

Note that the system obtained in example 7 is slightly different from the system obtained in example 6. Indeed transition relations of systems obtained by applying a property-based policy are not constrained when a request is applied on a non-secure state, while transition relations of systems obtained by applying a rule-based policy allow to apply a request only if the conditions specified by the policy are satisfied. The following proposition states a result about executions of systems constrained by a rule-based policy.

Proposition 4. *1.* $Exec\left(\lfloor\mathbb{S}\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right) \subseteq Exec\left(\lfloor\mathbb{S},\star\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right)$

2. If $\mathfrak{P}$ *satisfies the switching property according to* $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ *and* $\mathscr{I}$ *and if:*

$$\begin{array}{l}\forall l \in L\ \forall \sigma_1,\sigma_2 \in \Sigma \quad \sigma_1 \xrightarrow{l}_{\delta} \sigma_2 \\ \Rightarrow \left(\left(\exists (R,d) \in \mathcal{R}\times\mathcal{D}\ \ [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}}\right) \vee [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} = [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}}\right)\end{array}$$

then $Exec\left(\lfloor\mathbb{S},\star\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right) \subseteq Exec\left(\lfloor\mathbb{S}\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right)$.

Furthermore, we prove that each reachable state of a system constrained by $\mathfrak{P}$ can been obtained in a secure way according to the policy.

Proposition 5. *If* $(c,\sigma) \in \Gamma\left(\lfloor\mathbb{S}\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right)$ *or* $(c,\sigma) \in \Gamma\left(\lfloor\mathbb{S},\star\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right)$, *then* $\exists (c,A_0) \in \mathscr{I}\ \exists \overrightarrow{(R,d)} \in (\mathcal{R}\times\mathcal{D})^{\star}\ A_0 \xrightarrow{\overrightarrow{(R,d)}}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathfrak{P},c}{}^{\star} [\![\sigma]\!]_{I^{\mathbb{A}}_{\Sigma}}$.

Here again, defining a system from a rule-based policy $\mathfrak{P}$ can be done by considering $\mathcal{R}$ together with its semantics $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$, and a subset $\mathscr{I}$ of $\mathcal{C}\times\mathbb{A}$, and by applying $\mathfrak{P}$ (according to the interpretation $I^{\mathbb{A}}_{\mathbb{A}}$ such that $[\![A]\!]_{I^{\mathbb{A}}_{\mathbb{A}}} = A$) on the system $(\mathbb{A},\mathbb{A}^0,\mathcal{R}\times\mathcal{D},[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}})$, where $\mathbb{A}^0 = \{A \in \mathbb{A} \mid \exists (c,A) \in \mathscr{I}\}$. For example, from the policy $\mathfrak{P}_{\mathsf{hru}}$, the set $\mathscr{I} = \mathcal{C}_{\mathsf{hru}}\times\{\emptyset\}$, and the set $\mathcal{R}^{ac}$ together with its semantics $[\![\mathcal{R}^{ac}]\!]^{\mathcal{D}}_{\mathbb{A}_{ac}}$, we can define the system obtained in example 7 by applying $\mathfrak{P}_{\mathsf{hru}}$ on $\mathbb{S}_{ac}$. Of course, the transition relation δ can also be constrained by defining a relation such that:

$$\delta \subseteq \left\lfloor [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}} \right\rfloor^{I^{\mathbb{A}}_{\mathbb{A}}}_{\mathfrak{P},\mathcal{C},[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}$$

Equivalence results. We state here equivalence results about systems constrained by, or defined from, policies obtained with the constructions defined above. More precisely, we characterize the assumptions under which these systems have the same executions.

Proposition 6. *Let* $\mathbb{P} = (\mathbb{A},\mathcal{C},\Vdash)$ *be the* $(\mathfrak{P},[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I})$*-policy.*

1. $Exec\left(\lfloor\mathbb{S}\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right) \subseteq Exec\left(\lfloor\mathbb{S},\star\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}}\right) \subseteq Exec\left(\lfloor\mathbb{S}\rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C}\right)$
2. *If the three following properties hold:*

(a) $\forall (c, A_0) \in \mathscr{I}\ \forall A \in \mathbb{A}\ \forall \overrightarrow{(R,d)} \in (\mathcal{R} \times \mathcal{D})^\star$

$$\left(A_0 \xrightarrow{\overrightarrow{(R,d)}}{}^{\star}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}, \mathfrak{P}, c} A \wedge (\exists \sigma_0 \in \Sigma^0\ A = [\![\sigma_0]\!]_{I^{\mathbb{A}}_{\Sigma}}) \right) \Rightarrow (c, A) \in \mathscr{I}$$

(b) $\forall l \in L\ \forall \sigma_1, \sigma_2 \in \Sigma \quad \sigma_1 \xrightarrow{l}_{\delta} \sigma_2$

$$\Rightarrow \left(\left(\exists (R,d) \in \mathcal{R} \times \mathcal{D}\ [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \right) \vee [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} = [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \right)$$

(c) $\mathfrak{P}$ *satisfies the switching property according to* $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ *and* $\mathscr{I}$

then $Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} \right) \subseteq Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right) = Exec\left(\lfloor \mathbb{S}, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right)$.

Proposition 7. *Let* $\mathbb{S} = (\Sigma, \Sigma^0, L, \delta)$, $\mathfrak{P} = (\mathbb{A}, \mathcal{C}, \mathcal{R}, \mathcal{D}, \Vdash)$ *be the* $(\mathbb{P}, [\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}})$*-policy,* $C \subseteq \mathcal{C}$*, and* $\mathscr{I}$ *be the set* $\{(c, A) \mid c \Vdash A \wedge c \in C \wedge \exists \sigma \in \Sigma^0 \mid [\![\sigma]\!]_{I^{\mathbb{A}}_{\Sigma}} = A\}$.

1. *If* $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$ *is deterministic, then:*

$$Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right) \subseteq Exec\left(\lfloor \mathbb{S}, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right) \subseteq Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} \right)$$

2. *If the following property holds:*

$$\forall l \in L\ \forall \sigma_1, \sigma_2 \in \Sigma \quad \sigma_1 \xrightarrow{l}_{\delta} \sigma_2$$
$$\Rightarrow \left(\left(\exists (R,d) \in \mathcal{R} \times \mathcal{D}\ [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} \xrightarrow{(R,d)}_{[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}} [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \right) \vee [\![\sigma_1]\!]_{I^{\mathbb{A}}_{\Sigma}} = [\![\sigma_2]\!]_{I^{\mathbb{A}}_{\Sigma}} \right)$$

then $Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathbb{P},C} \right) \subseteq Exec\left(\lfloor \mathbb{S} \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right) \subseteq Exec\left(\lfloor \mathbb{S}, \star \rfloor^{I^{\mathbb{A}}_{\Sigma}}_{\mathfrak{P},C,[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}},\mathscr{I}} \right)$.

5 Related works

As we said, there exists now a large collection of literature describing how to define and to reason about policies. Without being exhaustive, we give in table 1 a classification of some of the most representative of the existing approaches.
Rule-based approaches. In [1], C-Datalog programs are used to specify some entities (subjects, authorizations, etc), their structure (hierarchies, roles, etc) and the relationships existing between them: the meaning of a policy is based on a stable semantics model of C-Datalog programs. Like in [15], such development aims to define an architecture within a particular authorization language. In [2, 3], an access control policy is defined as a set authorizations: a policy is viewed as a configuration together with its semantics and only free policies can be expressed. Such framework is used to compose policies. In [5], a more general approach to compose policies is introduced and is based on the Belnap logic, used to resolve conflicts and unspecified answers. In [7], in the context of access control, a clear distinction is done between configurations and authorizations, but security targets are not introduced in an explicit way: a policy is defined by a set of configurations, a set of operations allowing to modify configurations, and a relation characterizing correct access judgements according to

	Distinction betwwen $\mathbb{A}$ and $\mathcal{C}$	Approach	Def. of $[\![\mathcal{R}]\!]^{\mathcal{D}}_{\mathbb{A}}$	Def. of a system from the policy	Application of policies over independent systems	$\mathcal{D}$
[1–3, 13, 18]	no	rules	no	no	no	$\{\text{yes}, \text{no}\}$
[4, 9]	no	rules	yes	yes	no	arbitrary
[5]	no	rules	no	no	no	$\{\text{yes}, \text{no}, \top, \bot\}$
[6, 7]	yes	rules	no	no	no	$\{\text{yes}, \text{no}\}$
[8]	no	rules	yes	yes	no	$\{\text{yes}, \text{no}\}$
[12]	yes	property	yes	yes	no	arbitrary
[17]	no	property	yes	yes	no	$\{\text{yes}, \text{no}\}$

Table 1. Related works

configurations. Such an approach aims to compare policies in term of expressive power of the administrative language allowing to transform configurations and can only be used for free policies. A similar approach can be found in [6], where security engineering of lattice-based policies is considered. In [18], a finer comparison mechanism for rule-based policies (in term of expressive power of the administrative language) is introduced. In [13], a rule-based policy is defined as a conjuntion of first-order logic formula, and the authors characterize fragments of logic for which $\Vdash$ is decidable. In [8], rule-based policies are expressed as Datalog programs and semantics of requests is considered. Such an approach is similar to the one introduced in section 4 to define a system from a rule-based policy. However, the system and the policy are defined over the same vocabulary. In [9], the authors represent rule-based access control policies as rewriting systems and use such an approach to compose policies. In [4], this approach is modified in order to make a clear distinction between the policy and the semantics of requests: such a development is close to the approach introduced in section 4 to define a system from a rule-based policy but the system and the policy share the same "actions" and are defined over the same signature. This framework is used to check security properties (such as confidentiality or integrity) *via* a notion of morphisms between environments (close to our notion of interpretation).

Property-based approaches. Property-based approaches are simpler than rule-based approaches and few generic frameworks allow to define them: defining a property-based policy can be easily done without a framework. A framework becomes useful when comparing or composing policies, or when dealing with a particular class of properties (information flows, etc). Hence, many policies are defined by following the property-based approach in an implicit way. The most famous property-based policy is the Bell and LaPadula policy [16], based on the MAC and MAC$\star$ properties over sets of accesses. These developments specify security properties and define a transition system, which preserves these properties: this corresponds to the definition of a relation $\Vdash$ characterizing secure targets together with the (direct) definition of a secure system from the policy. Furthermore, note that many access control policies [1, 15, 2, 3, 5, 7, 18, 13, 8] are defined in term of sets of authorizations and could also be viewed as property-based poli-

cies: in this context, authorizations can be viewed as (parts of) secure targets, as well as granted access requests. In [12], property-based policies are defined and a comparison mechanism (based on simulation of the executions of secure systems obtained from policies) is introduced. A policy is defined by a predicate Ω over a set of states, describing both the targets and the configurations, and two mappings are introduced to distinguish targets and configurations: given a state σ, $\Lambda(\sigma)$ (resp. $\Upsilon(\sigma)$) denotes the target (resp. the configuration), and we have $\Upsilon(\sigma) \Vdash \Lambda(\sigma) \Leftrightarrow \Omega(\sigma)$. In [17], property-based policies are introduced and the framework is used to check that executions of a system are correct according to the policy. Other works aiming to formally characterize security properties preserved by executions of systems can be found in [10, 11].

6 Conclusion

Security of information systems has become a well-established field of computer science. Hence many approaches dealing with security have been developed. In this paper, we have identified the components involved in these approaches and the role they play in the definition of a policy, a system and a secure system. Two main approaches have been considered here: the property-based approach is an abstract way to specify the security properties we want to enforce, while the rule-based approach specifies which actions are granted. Hence, rule-based policies allows a fine control over executions of a system that cannot always be ensured by a property-based policy. However, in most of cases, rule-based policies one can find in the literature satisfies the switching property and can be expressed as property-based policies. Moreover, it should be notice that when targets contain information about the past, defining property-based policies that aim to control sequences of actions becomes possible. In any case, rule-based and property-based approaches are not equivalent, and our results allow to enlighten differences between the two approaches, by characterizing some properties the policies have to satisfy to be equivalent, and by characterizing some properties the policies and the systems have to satisfy in order to have the same executions. Thanks to this study, it becomes possible to relate the different existing approaches, thus allowing to reuse policies in a particular context even if these policies have been defined by following another approach. Hence, this paper provides a generic formal framework in which many security policies and systems can be specified, implemented and proved correct according to some security properties. We think that using formal specifications is largely beneficial. Indeed, in the literature, one can find papers presenting a particular security mechanism through examples without any formalisation (or generalisation) of the concepts involved in the mechanism. Of course, such papers are very useful to understand how a particular mechanism works but they provide little help to implement it. We also think that genericity is important: it allows to have a common formalism (in which policies are described) thus allowing to characterize particular classes of policies and systems (such as free policies), to compare, to define operations (such as composition) and to reason in a generic way about policies and sys-

tems. The definition of such a framework contributes to a better understanding of what is a security policy. This framework has been successfully used to define and analyse classical access control policies (HRU, RBAC, Bell & LaPadula, Chinese Wall, Trust Management).

References

1. E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. In *SACMAT*, pages 41–52, 2001.
2. P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. A modular approach to composing access control policies. In *ACM Conf. on Computer and Communications Security*, pages 163–173, 2000.
3. P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Trans. on Inf. and Syst. Security*, 5(1):1–35, 2002.
4. T. Bourdier, H. Cirstea, M. Jaume, and H. Kirchner. Rule-based Specification and Analysis of Security Policies. 5th International Workshop on Security and Rewriting Techniques, SECRET2010.
5. G. Bruns and M. Huth. Access-control policies via Belnap logic: Effective and efficient composition and analysis. In *Proc. of the 21st IEEE Computer Security Foundations Symposium, CSF 2008*, pages 163–176. IEEE Computer Society, 2008.
6. C. Bryce. Security engineering of lattice-based policies. In *Proc. of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.
7. A. Chander, J.C. Mitchell, and D. Dean. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundation Workshop CSFW*, pages 27–43. IEEE Comp. Society Press, 2001.
8. D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In *Automated Reasoning, Third Int. Conf., IJCAR 2006, Proceedings*, volume 4130 of *LNCS*, pages 632–646. Springer, 2006.
9. D. J. Dougherty, C. Kirchner, H. Kirchner, and A. Santana de Oliveira. Modular access control via strategic rewriting. In *ESORICS 2007, Proc.*, volume 4734 of *LNCS*, pages 578–593. Springer, 2007.
10. S. Gürgens, P. Ochsenschläger, and C. Rudolph. Abstractions preserving parameter confidentiality. In *ESORICS 2005*, pages 418–437. LNCS, Springer-Verlag, 2005.
11. S. Gürgens, P. Ochsenschläger, and C. Rudolph. On a formal framework for security properties. *Computer Standards & Interfaces*, 27(5):457–466, 2005.
12. L. Habib, M. Jaume, and C. Morisset. Formal definition and comparison of access control models. *J. of Information Assurance and Security*, 4(4):372–381, 2009.
13. J. Y. Halpern and V. Weissman. Using first-order logic to reason about policies. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.
14. M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the ACM*, 19:461–471, 1976.
15. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):474–485, June 1997.
16. L.J. LaPadula and D.E. Bell. Secure Computer Systems: A Mathematical Model. *Journal of Computer Security*, 4:239–263, 1996.
17. J. Ligatti, L. Bauer, and D. Walker. Run-time enforcement of nonsafety policies. *ACM Trans. Inf. Syst. Secur*, 12(3), 2009.
18. M.V. Tripunitara and N. Li. Comparing the expressive power of access control models. In *11th ACM Conf. on Computer and Communications Security*, 2004.