Quantitative Evaluation of Enforcement Strategies
Position Paper

Vincenzo Ciancia[1], Fabio Martinelli[2], Matteucci Ilaria[2](✉), and Charles Morisset[3]

[1] CNR-ISTI, Via Moruzzi 1, 56124 Pisa, Italy
vincenzo.ciancia@isti.cnr.it
[2] CNR-IIT, Via Moruzzi 1, 56124 Pisa, Italy
{fabio.martinelli,matteucci.ilaria}@iit.cnr.it
[3] School of Computing Science, Newcastle University, NE17RU, Newcastle, UK
charles.morisset@ncl.ac.uk

Abstract. A *security enforcement mechanism* runs in parallel with a system to check and modify its run-time behaviour, so that it satisfies some security policy. For each policy, several enforcement strategies are possible, usually reflecting trade-offs one has to make to satisfy the policy. To evaluate them, multiple dimensions, such as security, cost of implementation, or cost of attack, must be taken into account. We propose a formal framework for the quantification of enforcement strategies, extending the notion of *controller processes* (mimicking the well-known edit automata) with weights on transitions, valued in a semiring.

Keywords: Enforcement mechanisms · Quantitative process algebra · Semiring

1 Introduction

Security is often regarded as a binary concept, as it usually strictly depends on satisfaction of a boolean policy. However, in a broader sense, security has several dimensions e.g., secrecy, anonymity, enforceability, availability, risk, trust. These interesting features give rise to a multi-dimensional solution space for the construction of a "secure program". Functional requirements add to the picture costs, execution times, rates, etc. All these dimensions make it meaningless to talk about a "secure" program, shifting the focus to the definition of a globally optimal solution, which easily fails to exist.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grants no 256980 (NESSoS) and no 295354 (SESAMO). This work has been also partially supported by the TENACE PRIN Project (no 20103P34XC) funded by the Italian Ministry of Education, University and Research.

In this work, we consider security as a quantitative, multi-dimensional measure of a system, and investigate possible answers to the question of what it means to enforce a security policy in this new setting. Our main actors will be *controllers* that constrain *targets* to obey to policies, using *enforcement strategies*. The ultimate goal of the proposed research direction is to define *quantitative evaluation of enforcement strategies*, that would provide analysis tools to compare and select different controllers, according to several metrics. Instead of asking if a controller enforces a policy or not, one can ask if a controller is the *optimal* one for a certain combination of metrics. The plethora of dimensions demands for a parametric approach. We address this aspect by adopting *semirings*, that are well-known domains for optimization problems (see e.g., [1]) and permit multi-dimensional valuations by composition.

Summing up, beyond the state of the art (Sect. 2), in this paper we make a distinction between monitors and controllers. Monitors (Sect. 4) associate quantities to an existing system without changing its behaviour. Controllers (Sect. 5) modify the behaviour of a target, using *control actions* for suppressing or inserting possible incorrect actions. In Sect. 6 we propose a formal approach to evaluate and compare controllers in the quantitative, multi-dimensional setting.

2 State of the Art

Runtime enforcement is a well-studied problem [2–4], in particular with security automata [2], designed to prevent bad executions, and edit automata [5], that can suppress, insert or replace observed actions. Concurrent languages (e.g., process algebras) can also model both the target and the controlled system within the same formalism [3,6]. As a prototypical example, we chose the process algebra GPA [7], featuring CSP-style synchronization, with actions weighted over a semiring. We add to it control operators in the style of edit automata, in order to study enforcement strategies from the quantitative standpoint. Compared to the existing literature, our work identifies an abstract approach to quantitative and multi-dimensional aspects of security, by introducing semirings. The quest for a unifying formalism is witnessed by the significant amount of inhomogeneous work in quantitative notions of security and enforcement.

The problem of finding an optimal control strategy is considered in [8] in the context of software monitoring, taking into account rewards and penalties, and can be solved using a Markov Decision Process [9]. Bielova and Massacci propose in [10] a notion of *edit distance* among traces, that extends to an ordering over controllers.

Metrics can be used for evaluating the correctness of a mechanism, such as *More Corrective*, *Permissiveness*, *Memory Use*, and *Versatility* [11]. Additionally, mechanisms can be compared with respect to their *cost* [12]. In this work, we follow some intuitive leads from [13] to move from qualitative to quantitative enforcement, and generalise that idea using semirings. We also consider the possibility for a controller not to be correct, i.e., to allow for some violations of the policy. Such a possibility is quantified over traces in [14] for non-safety policies, where a controller cannot be both correct and fully transparent. In [15], the authors use a notion of *lazy* controllers, which only check the security of a system at some points in time, proposing a probabilistic quantification of the expected risk.

3 Quantitative Processes

In a *quantitative process*, observable transitions are labelled with some quantity, denoting a cost or a benefit associated to a step in the behaviour of a system. We use *semirings* to model two fundamental modes of composing observable behaviour, either by combination of different traces, or by sequential composition. As a syntax to describe such behaviour, we adopt GPA from [7]. We first provide the definition of a semiring.

Definition 1. *A* semiring $\mathbb{K} = (K, +, *, \mathbf{0}, \mathbf{1})$ *consists of a set K with two binary operations $+, *$, and two constants $\mathbf{0}, \mathbf{1}$, such that $+$ is associative, with neutral element $\mathbf{0}$; $*$ is associative, with neutral and absorbing elements $\mathbf{1}, \mathbf{0}$; $*$ distributes over $+$.*

Examples of semirings are natural numbers, the positive real numbers, boolean algebras, that may denote, e.g., time, costs, lattices of values. Semirings have a partial order $\sqsubseteq$, such that $k_1 \sqsubseteq k_2$ if, and only if $k_1 + k_2 = k_2$. Intuitively, $\sqsubseteq$ indicates *preference*, that is, $k_1 \sqsubseteq k_2$ can be read as k_2 is "better" than k_1. Sometimes, the $+$ operation is idempotent, and it extends to an operation $\sum_{\{S\}}$ defined over an arbitrary, possibly infinite subset S of K. The well-known C-semirings [1] are of this form. For the sake of simplicity, when needed, we silently assume that this is the case.

The cartesian product of semirings is a semiring; thus, multi-dimensional notions of cost can be modelled. The partial order of values in the product does not prioritize dimensions. Further composite semirings exist, such as the lexicographic semiring, the expectation semiring, etc. When the specific composition operator is not relevant, we shall indicate a composite semiring by $\mathbb{K}_1 \odot \ldots \odot \mathbb{K}_n$.

Process algebras are simple languages with precise mathematical semantics, tailored to exhibit and study specific features of computation. Typically, a *process* P, specfied by some syntax, may non-deterministically execute several *labelled transitions* of the form $P \xrightarrow{a} P'$, where a is an observable effect and P' is a new process. In quantitative process algebras, transitions are labelled by pairs (a, x) where x is a quantity associated to the effect a. We now define *Generalized Process Algebra* (GPA).

Definition 2. *The set $\mathcal{L}$ of* agents*, or processes, in GPA over a countable set of transition labels Act and a semiring $\mathbb{K}$ is defined by the grammar*

$$A ::= 0 \mid (a, k).A \mid A + A \mid A \|_S A \mid X$$

where $a \in Act$, $k \in K$, $S \subseteq Act$, and X belongs to a countable set of process variables*, coming from a system of co-recursive equations of the form $X \triangleq A$. We write* $\mathsf{GPA}[\mathbb{K}]$ *for the set of GPA processes labelled with weights in $\mathbb{K}$.*

Process 0 describes inaction or termination; $(a, k).A$ performs a with *weight* k and evolves into A; $A+A'$ non-deterministically behaves as either A or A'; $A \|_S A'$ describes the process in which A and A' proceed concurrently and independently on all actions which are not in S, and synchronize over actions in S.

As we are dealing with *run-time* enforcement, we work with *traces*, or paths, of processes. A *path* is a sequence $(a_1, k_1) \cdots (a_n, k_n)$, and we call $\mathcal{T}(A)$ the set of paths rooted in A. Given a path $(a_1, k_1) \cdots (a_n, k_n)$, we define its *label* $l(t) = a_1 \cdots a_n$, and its *run weight* $|t| = k_1 * \ldots * k_n \in K$. Finally, the *valuation* of a process A is given by $[\![A]\!] = \sum_{\{t \in \mathcal{T}(A)\}} |t|$.

4 Quantitative Monitor Operators

A system, hereafter named *target*, does not always come with the quantities we are interested in evaluating, and might even be not labelled at all. Hence, in the most general case, the security designer must provide a *labelling function* $\lambda : \mathsf{GPA}[\mathbb{K}_1] \to \mathsf{GPA}[\mathbb{K}_2]$, such that given any process A labelled in $\mathbb{K}_1$, $\lambda(A)$ represents the process A labelled with a quantity in $\mathbb{K}_2$. A simple example is the function λ_v, which assign any transition with the value $v \in \mathbb{K}_2$, thus erasing any previous quantity.

In practice, a *monitor* often measures a particular aspect, by probing the system and indicating the weight of each operation. In terms of security, a monitor is usually passive, i.e., it does not effectively modify the behaviour of the considered target, and thus does not prevent violation of a security policy. On the other hand, a *controller* is able to modify the behaviour of a target in order to guarantee security requirements. A security monitor and a security controller are often merged into a single entity, responsible both for deciding whether an action would violate the policy and what corrective action should be taken if necessary. We propose to make an explicit distinction between these two processes and to extend the monitoring to measures other than security. In this section we investigate quantitative monitors. Controllers are detailed in Sect. 5.

Intuitively, a monitor measures a quantity not already present in the monitored target. Since the target might be already equipped with some quantities, coming for instance from another monitor, we need to *merge* the quantities from the monitor with those of the target. Given a process A labelled with $\mathbb{K}$, a process A' labelled with $\mathbb{L}$ and a composition operator $\odot$, we write $A \odot A'$ for the merged process, defined as:

$$\frac{A \overset{a,k}{\rightarrow} A_1 \quad A' \overset{a,l}{\rightarrow} A'_1}{A \odot A' \overset{a,k \odot l}{\rightarrow} A_1 \odot A'_1}$$

A merged process can only move on when both of its components can move on with the same action. We are now able to define a monitor, which is a process that can be composed with a target without affecting its behaviour.

Definition 3. *Given a composition operator $\odot$ and a process A, a process M is a monitor for A if and only if $\{l(t) \mid t \in \mathcal{T}(A \odot M)\} = \{l(t) \mid t \in \mathcal{T}(A)).$*

Given any process A labelled with $\mathbb{K}_1$, any monitor M for A labelled with $\mathbb{K}_2$ and any composition operator $\odot$, we can define the labelling function $\lambda : \mathsf{GPA}[\mathbb{K}_1] \to \mathsf{GPA}[\mathbb{K}_1 \odot \mathbb{K}_2]$ as $\lambda(A) = A \odot M$.

Example 1. Let us define an energy monitor using $\mathbb{K}_C = \langle \mathbb{R}_0^+, min, +, +\infty, 0\rangle$ for the alphabet $\Sigma = \{a, b\}$, such that the action a consumes 3 units, and the action b consumes $2n$ units, where n is the number of times b has been performed (i.e., b has an increasing energy cost). Hence, for $n > 0$, we define the monitor:

$$M_n = (a, 3).M_n + (b, 2n).M_{n+1}$$

For instance, the process $A = a.b.b.a.b$ can be monitored with

$$A \odot M_1 = (a, 3).(b, 2).(b, 4).(a, 3).(b, 6)$$

The valuation of the monitored process corresponds to the total energy consumed, i.e., $[\![A \odot M_1]\!] = 18$. Similarly, the monitored process of $B = a + b$ is $B \odot M_1 = (a, 3) + (b, 2)$, and its valuation $[\![B \odot M_1]\!] = 2$, since the valuation returns the best quantity.

Clearly, finer-grained approaches can be used to monitor a security policy. Note that a monitor is only one possible way to build a labeling function λ. Although monitors are expressive enough for the examples, in this paper, we consider more complex labeling functions may also be of interest.

5 Quantitative Control Operators

The role of the monitor is to *observe* the actions from a target, and not to *prevent* a *target* system from performing any. For instance, given a policy P, a monitor can observe the target actions, labelling them with *true* when they obey to the policy P or *false* when they violate P. On the contrary, a *controller* can decide not only to accept but also to change target traces, which result in a *controlled process* $E \triangleright F$, where F denotes the target system, following the semantics in Fig. 1.

Intuitively speaking, each rule corresponds to a different controlling behaviour. The alphabets of E, F, and of the resulting process $E \triangleright F$ are different, as E may perform *control actions* that regulate the actions of F, and $E \triangleright F$ may perform internal actions, denoted by τ, as a consequence of suppression. Let *Act* be the alphabet of (the GPA describing) F, and let $\{a, \boxplus a.b, \boxminus a\}$, for $a, b \in Act$, be the alphabet of E, respectively denoting *acceptance*, *suppression*, and *insertion*; the alphabet of $E \triangleright F$ is $Act \cup \{\tau\}$.

$$\frac{E \xrightarrow{a,k} E' \quad F \xrightarrow{a,k'} F'}{E \triangleright F \xrightarrow{a,k*k'} E' \triangleright F'}\ (\mathrm{A}) \qquad \frac{E \xrightarrow{\boxminus a,k} E' \quad F \xrightarrow{a,k'} F'}{E \triangleright F \xrightarrow{\tau,k*k'} E' \triangleright F'}\ (\mathrm{S}) \qquad \frac{E \xrightarrow{\boxplus a.b,k} E' \quad F \xrightarrow{a,k'} F'}{E \triangleright F \xrightarrow{b,k} E' \triangleright F}\ (\mathrm{I})$$

Fig. 1. MLTS rules for quantitative control operators.

The *acceptance rule* (A) constrains the controller and the target to perform the same action, in order for it to be observed in the resulting behaviour; the observed weight is the product of those of the controller and the target. Given two processes A and B, the semantics of truncation is equivalent to that of CSP-style parallel composition of A and B, where synchronisation is forced over all actions of the two processes.

The *suppression rule* (S) allows the controller to hide target actions. The target performs the action, but the controlled system does not, and the observed result is a τ action, with the weight given as the product of the suppressing and the target action.

The *insertion rule* (I) describes the capability of correcting some bad behaviour of the target, by inserting another action in its execution trace. The weight of insertion is only the weight provided by the controller; this accounts for the fact that the target does not perform any action, but rather stays in its current state, as in [5].

Interestingly, monitoring a target F inside a controlled system $E \triangleright F$ or monitoring $E \triangleright F$ leads to different valuations. For instance, monitoring F for a policy P will associate with $false$ each trace where F tried to violate P, even though the action was corrected by E; on the other hand, monitoring $E \triangleright F$ for P associates each trace with $true$ if E correctly enforces P. In other words, the nesting of monitors impacts the valuation of processes, and we therefore introduce the notion of *matching operator* $\bowtie$:

$$E \bowtie F = \lambda_T(\lambda_E(E) \triangleright \lambda_F(F))$$

where λ_E labels the controller, λ_F labels the target and λ_T labels the controlled target.

For instance, we can consider the following evaluation strategies:

$$\begin{array}{ll} E \bowtie_D F = \lambda_{true}(E) \triangleright (F \odot M_P) & E \bowtie_P F = (E \triangleright F) \odot M_P \\ E \bowtie_C F = (E \odot M_c) \triangleright (F \odot M_1) & E \bowtie_G F = M_P \odot_L (E \bowtie_C F) \end{array}$$

where M_P stands for the monitor for the policy P, $\bowtie_D$ detects policy violations, even if they are corrected by the controller, $\bowtie_P$ monitors the satisfaction of the policy by the controlled target, $\bowtie_C$ monitors the energy of both the controller and the target, and $\bowtie_G$ defines a lexicographic measure of the cost and the satisfaction of the policy.

6 Ordering Controller Strategies

The matching operator defined provides a simple way to compare controllers, for any considered semiring.

Definition 4. *Given a target F and a matching operator $\bowtie$, a controller E_2 is* better *than a controller E_1 with respect to F, and in this case, we write $E_1 \sqsubseteq_{\bowtie,F} E_2$, if and only if $[\![E_1 \bowtie F]\!] \sqsubseteq [\![E_2 \bowtie F]\!]$. Furthermore, we write*

$E_1 \sqsubseteq_{\bowtie} E_2$ and say that E_1 is always better than E_2 if and only if $E_1 \sqsubseteq_{\bowtie,F} E_2$, for any target F. Finally, if $E_1 \sqsubseteq_{\bowtie} E_2$ and there exists at least one target F such that $[\![E_1 \bowtie F]\!] \sqsubset [\![E_2 \bowtie F]\!]$, we write $E_1 \sqsubset_{\bowtie} E_2$.

This definition does not directly depend on the semiring used to quantify the controlled target, and it is therefore possible to use the same definition to say that a controller is better than another one with respect to a security monitor, a cost monitor or any other measure. Note that since each individual trace can be represented as a target, $E_1 \sqsubseteq_{\bowtie} E_2$ implies that the valuation of E_1 should be lower than that of E_2 for every possible trace.

Example 2. Let us extend the example described in Sect. 4, such that we have now three actions $\{a, b, c\}$, and a policy P stating that any trace should start with at least one action b. Now, consider the four following controllers:

$$\begin{array}{ll} E_1 = a.E_1 + b.E_1 + c.E_1 & E_2 = \boxminus a.E_2 + b.E_1 + c.E_2 \\ E_3 = \boxminus a.E_3 + b.E_1 + \boxminus c.E_3 & E_4 = \boxplus a.b.E_1 + b.E_1 + \boxplus c.b.E_1 \end{array}$$

Intuitively, E_1 accepts all actions, E_2 suppresses all initial actions a, but accepts c, E_3 suppresses both actions a and c, and E_4 inserts a b before any initial a or c. As soon as an action b is performed, all processes are equivalent to E_1, and accept all actions.

Since E_3 and E_4 are sound, we have $[\![E_3 \bowtie_P F]\!] = [\![E_4 \bowtie_P F]\!] = \mathit{true}$, for any target F. In addition, given any target F such that $[\![E_2 \bowtie_P F]\!] = \mathit{false}$, we also have $[\![E_1 \bowtie_P F]\!] = \mathit{false}$. Since there are also targets F such that $[\![E_2 \bowtie_P F]\!] = \mathit{true}$ and $[\![E_1 \bowtie_P F]\!] = \mathit{false}$, we have $E_1 \sqsubset_{\bowtie_P} E_2 \sqsubset_{\bowtie_P} E_3 \equiv_{\bowtie_P} E_4$, where $\equiv_{\bowtie}$ is the equivalence relation induced by the partial order $\sqsubseteq_{\bowtie}$. In other words, E_3 and E_4 are maximal, and E_1 is strictly worse than E_2.

However, it is worth observing that E_1 is not the worst possible controller. Indeed, E_1 leaves unchanged the correct traces of F, meaning that there exists some targets F such that $[\![E_1 \bowtie_P F]\!] = \mathit{true}$. The worst controller always outputs incorrect traces, even when the target is correct. For instance, we can define the controller $E_0 = \boxplus b.a.E_1 + a.E_0 + c.E_0$, which satisfies $[\![E_0 \bowtie_P F]\!] = \mathit{false}$, for any target F, and therefore $E_0 \sqsubset_{\bowtie_P} E_1$.

In some cases, controllers can be incomparable. In the previous example, the controller that only suppresses bad actions a is incomparable with the one that only suppresses bad actions c. Furthermore, the choice of the controlling operators can have an impact on the overall evaluation. For instance, if one cannot implement the controlling strategy E_3 because the action c is uncontrollable [16], i.e., cannot be suppressed or protected, then a security designer may prefer to choose E_2 over E_1, and over E_0 even if they are incorrect. The controllers E_3 and E_4 are equivalent with respect to $\bowtie_P$, since they are both correct, and, if policy satisfaction is the only criterion, a security designer might choose either. However, other dimensions can easily be included within our framework, with the intuition that the more accurate is the quantification of the controlled system, the more informed is the security designer to choose a controller.

Example 3. In order to compare the previous controllers E_3 and E_4, let us consider the extended cost monitor $M_c = (x, 0).M_c + (\boxminus y, 1).M_c + (\boxplus w.v, 2).M_c$ with the matching operator $\bowtie_G$. First, it is worth observing that since we use the lexicographic ordering, the relations $E_0 \sqsubset_{\bowtie_G} E_1 \sqsubset_{\bowtie_G} E_2 \sqsubset_{\bowtie_G} E_3$ and $E_0 \sqsubset_{\bowtie_G} E_1 \sqsubset_{\bowtie_G} E_2 \sqsubset_{\bowtie_G} E_4$ still hold. However, E_3 and E_4 are no longer equivalent, and as a matter of fact, they become incomparable. Indeed, consider the target $F_1 = a$: we have $[\![E_3 \bowtie_G F_1]\!] = (true, 1)$ and $[\![E_4 \bowtie_G F_1]\!] = (true, 2)$, meaning that $E_4 \sqsubset_{\bowtie_G, F_1} E_3$. On the other hand, given the target $F_2 = a.a.a$, we have $[\![E_3 \bowtie_G F_1]\!] = (true, 3)$ and $[\![E_4 \bowtie_G F_1]\!] = (true, 2)$, meaning that $E_3 \sqsubset_{\bowtie_G, F_2} E_4$, and therefore that E_3 and E_4 are incomparable.

The previous example illustrates that, in general, there might not be a strictly best strategy. In some cases, it might be possible to define an *optimal* strategy, which is best *in average*. Another possibility is to prioritize one dimension over another (depending on the order of the components in the lexicographic order itself). According to which dimension is prioritized, we are able to classify controllers into categories. For instance, we have a *secure controller*, when the controllers are ordered based on their security or an *economical controller* when the priority is given to the dimension of cost.

7 Discussion – Future Work

Our framework allows a system designer to consider security as yet another dimension to measure. Instead of a binary classification between sound/unsound controllers, we provide a finer-grained ordering, distinguishing between different degrees of soundness.

The valuation of processes is currently done on the best-case scenario. It is not difficult to focus instead on the worst-case scenario, thus following a rather traditional, pessimistic approach to security. In the presence of an inverse to addition in the considered semiring, the two kind of valuations could be mixed. As we saw, some controllers are incomparable over the set of all targets. To improve the situation, one can consider small subsets of possible targets (e.g., typical behaviours, or even single use-cases).

Adding non-deterministic choice to the controller itself always improves security. Clearly, this characteristic is mostly of theoretical importance, but it raises the interesting question whether, given some quantities, there exists a deterministic maximal controller or not. For instance, if we only monitor security, we can build an optimal deterministic controller for any safety property, as described in the literature [5]. However, adding a notion of cost can lead to two incomparable deterministic controllers, which are strictly worse than their non-deterministic composition.

Finally, our notion of a security policy is just a boolean predicate, that could be specified by e.g., automata or logics. Predicates, and formulas specifying them, could also be quantitative by themselves, e.g., employing logics with valuations in a semiring, e.g., [17,18]. In this paper, we do not yet investigate this aspect of the framework; this is left as future work. In particular, we plan to use quantitative evaluation of security policies, specified by logic formulas, in order to extend previous work on automated verification and synthesis of (qualitative) controllers [19].

References

1. Bistarelli, S.: Semirings for Soft Constraint Solving and Programming. LNCS, vol. 2962. Springer, Heidelberg (2004)
2. Schneider, F.B.: Enforceable security policies. ACM TISSEC **3**(1), 30–50 (2000)
3. Martinelli, F., Matteucci, I.: Through modeling to synthesis of security automata. ENTCS **179**, 31–46 (2007)
4. Khoury, R., Tawbi, N.: Which security policies are enforceable by runtime monitors? a survey. Comput. Sci. Rev. **6**(1), 27–45 (2012)
5. Bauer, L., Ligatti, J., Walker, D.: Edit automata: enforcement mechanisms for run-time security policies. Int. J. Inf. Secur. **4**(1–2), 2–16 (2005)
6. Gay, R., Mantel, H., Sprick, B.: Service automata. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 148–163. Springer, Heidelberg (2012)
7. Buchholz, P., Kemper, P.: Quantifying the dynamic behavior of process algebras. In: de Luca, L., Gilmore, S. (eds.) PAPM-PROBMIV 2001. LNCS, vol. 2165, pp. 184–199. Springer, Heidelberg (2001)
8. Easwaran, A., Kannan, S., Lee, I.: Optimal control of software ensuring safety and functionality. Technical report MS-CIS-05-20, University of Pennsylvania (2005)
9. Martinelli, F., Morisset, C.: Quantitative access control with partially-observable markov decision processes. In: Proceedings of CODASPY '12, pp. 169–180. ACM (2012)
10. Bielova, N., Massacci, F.: Predictability of enforcement. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 73–86. Springer, Heidelberg (2011)
11. Khoury, R., Tawbi, N.: Corrective enforcement: a new paradigm of security policy enforcement by monitors. ACM Trans. Inf. Syst. Secur. **15**(2), 10:1–10:27 (2012)
12. Drábik, P., Martinelli, F., Morisset, C.: Cost-aware runtime enforcement of security policies. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 1–16. Springer, Heidelberg (2013)
13. Martinelli, F., Matteucci, I., Morisset, C.: From qualitative to quantitative enforcement of security policy. In: Kotenko, I., Skormin, V. (eds.) MMM-ACNS 2012. LNCS, vol. 7531, pp. 22–35. Springer, Heidelberg (2012)
14. Drábik, P., Martinelli, F., Morisset, C.: A quantitative approach for inexact enforcement of security policies. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 306–321. Springer, Heidelberg (2012)
15. Caravagna, G., Costa, G., Pardini, G.: Lazy security controllers. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 33–48. Springer, Heidelberg (2013)
16. Basin, D., Jugé, V., Klaedtke, F., Zălinescu, E.: Enforceable security policies revisited. In: Degano, P., Guttman, J.D. (eds.) POST 2012. LNCS, vol. 7215, pp. 309–328. Springer, Heidelberg (2012)
17. Lluch-Lafuente, A., Montanari, U.: Quantitative mu-calculus and ctl defined over constraint semirings. TCS **346**(1), 135–160 (2005)
18. Ciancia, V., Ferrari, G.L.: Co-algebraic models for quantitative spatial logics. ENTCS **190**(3), 43–58 (2007)
19. Martinelli, F., Matteucci, I.: A framework for automatic generation of security controller. Softw. Test. Verif. Reliab. **22**(8), 563–582 (2012)