There are Two Sides to Every Question
Controller Versus Attacker

Fabio Martinelli(✉), Ilaria Matteucci, and Francesco Santini

Istituto di Informatica e Telematica, IIT-CNR, Pisa, Italy
{Fabio.Martinelli,Ilaria.Matteucci,Francesco.Santini}@iit.cnr.it

Abstract. We investigate security enforcement mechanisms that run in parallel with a system; the aim is to check and modify the run-time behaviour of a possible attacker in order to guarantee that the system satisfies some security policies. We focus on a CSP-like quantitative process-algebra to model such processes. Weights on actions are modelled with semirings, which represent a parametric structure where to cast different metrics. The basic tools are represented by a quantitative logic and a model checking function. First, the behaviour of the system is removed from the parallel computation with respect to some security property to be satisfied. Secondly, what remains is refined in two formulas with respect to the given operator executed by a controller. The result describes what a controller has to do to prevent a given attack.

1 Introduction

Security is frequently in conflict with functional requirements, such as costs, execution times, and rates, as well as performance requirements of a system, making 100 % security an impossible or overly expensive goal to be accomplished. Therefore, the relevant question is not whether a system is secure, but rather how much security it provides under such "soft" constraints.

In addition, we can use security-oriented metrics, as the vulnerability exposure (the sum of known and unpatched vulnerabilities), the worst case loss (the maximum money-value of the damage/loss that could be inflicted), the data transmission exposure (the unencrypted data-transmission volume), or the detection performance (a measure of the effectiveness of the detection mechanisms implemented on the system) [8]. Instead of a plain yes/no answer, quantitative levels of security can express different degrees of protection, and allow a security expert to reason about the trade-off between security and conflicting requirements. Quantitative security analysis [19] has been already applied, *e.g.*, to name a few, for quantifying the side-channel leakage in cryptographic algorithms, for capturing the loss of privacy in statistical data analysis or information flows, and for quantifying security in anonymity networks.

Concurrent languages (*e.g.*, process algebras) are expressive enough to model a system, a controller, and an attacker within the same formalism. As a prototypical example, in this work we choose *Generalised Process Algebra* (*GPA*) [9]

The author is supported by MIUR PRIN 2010XSEMLC "Security Horizons".

C. Bodei et al. (Eds.): Degano Festschrift, LNCS 9465, pp. 304–318, 2015.
DOI: 10.1007/978-3-319-25527-9_20

featuring a CSP-style synchronisation; actions are weighted over a semiring algebraic-structure. In a quantitative process, transitions are labelled with some quantity, denoting a cost or a benefit associated with a step in the behaviour of a system. Indeed, the main actors on stage are the GPA behaviour-descriptions of *(i)* a system S, *(ii)* a controller C, and *(iii)* an attacker A, which all run in parallel and may synchronise on a set of action $L \subseteq Act$. S represents the correct behaviour of a (software) system, while A attempts to divert S from the expected path. A controller is implemented to correct/mitigate a threat against the system (*insertion*), suppress the impact of a threat (*suppression*), or ignore it (*acceptance*), i.e., $C \triangleright^{\mathbb{K}} A$. These actors are required to follow a plot, defined in terms of a formula ϕ in a c-semiring Hennessy-Milner Logic (*c-HM*) [25]. Such formula specifies the requested behaviour, and its evaluation corresponds to a semiring value, i.e., the amount of weight needed to follow that behaviour. Clearly, the overall cost depends on action-weights of S, C, as well as A, and, more in particular, on the action performed by C in order to restrain the behaviour of A (i.e., insertion, suppression, or acceptance).

Our main tool consists of a quantitative partial model-checking function (QPMC) we use to remove S from the scene, since its performance is not significant for our purposes: indeed we want to focus on the other two actors on the stage. Hence, the specification of S is moved from $S\|_L(C \triangleright^{\mathbb{K}} A)$ to ϕ, which becomes ϕ' consequently to the application of the QPMC function to ϕ with respect to S. The next and final step is the application of QPMC to refine ϕ' in a binary c-semiring Hennessy-Milner formula (*c-HM*^{2}) whose modalities represent the couple of action a_1 and a_2 that respectively represent the reaction of C and A: if A plays a_2, then C plays a_1.

In this way, we are able to identify the requested property ϕ on the shoulders of C and A, knowing exactly what C has to do for a given A. This corresponds to an amount t of weight demanded to satisfy ϕ: if t is the requested level of time-delay on the execution of S, when A introduces a delay then C has to react to A by performing some action with the aim to maintain t. "Does C exist or not exist? That is the question". Therefore, in this paper we find an answer to $\exists C\ \forall A\ S\|_L(C \triangleright^{\mathbb{K}} A) \models_t \phi$.

The main basic ideas behind this work are an advancement of what is proposed in [25], where, among other results, we propose a unidimensional *c-HM* logic and a similar QPMC function. The paper is organised as a five-act drama, following the Freytag's structure-pyramid.[1] Section 2 is our *exposition*: we introduce the necessary preliminary notions at the heart of our approach, i.e., semirings, GPA, and definitions for quantitative control-rules, and the related work. Then, Sect. 3 is the *rising action* part; it builds toward the point of greatest interest, an approach to security of the presented ingredients. Afterwards, we reach the *climax* in Sect. 4: there we show how to apply the QPMC function on control-rules. During the *falling action* (Sect. 5) we provide an example of our approach on the *Chinese Wall* policy, while Sect. 6 presents a *dénouement*, i.e., a resolution of the plot (conclusions and future work).

[1] Freytag, Gustav. *Die Technik des Dramas*. Hirzel, 1872.

2 Setting up the Scene

We start this section by introducing the algebraic formalism we adopt to represent quantitative metrics on which we evaluate countermeasures' behaviour in the following of the paper.

2.1 Semiring

Definition 1 (c-semiring [6]). *A c-semiring is a five-tuple* $\mathbb{K} = \langle K, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *such that* K *is a set,* $\mathbf{1}, \mathbf{0} \in K$*, and* $+, \times : K \times K \to K$ *are binary operators making the triples* $\langle K, +, \mathbf{0}\rangle$ *and* $\langle K, \times, \mathbf{1}\rangle$ *commutative monoids (semigroups with identity), satisfying (i) (distributivity)* $\forall a, b, c \in K. a \times (b + c) = (a \times b) + (a \times c)$*, (ii) (annihilator)* $\forall a \in A. a \times \mathbf{0} = \mathbf{0}$*, and (iii) (top element)* $\forall a \in K. a + \mathbf{1} = \mathbf{1}$.

The idempotency of $+$ leads to the definition of a partial ordering $\leq_K$ over the set K (K is a poset). Such partial order is defined as $a \leq_K b$ if and only if $a + b = b$, and $+$ becomes the *least upper bound* (*lub*) of the lattice $\langle K, \leq_K\rangle$. This intuitively means that b is "better" than a. As a consequence, we can use $+$ as an optimisation operator that always chooses the best available solution. Other properties can be derived on c-semirings [6]: (i) both $+$ and $\times$ are monotone over $\leq_K$, (ii) $\times$ is intensive (i.e., $a \times b \leq_K a$), and (iii) $\langle K, \leq_K\rangle$ is a complete lattice where $\mathbf{0}$ and $\mathbf{1}$ are its bottom and top elements, respectively.

Some examples of semiring instantiation are: *boolean* $\langle \{F, T\}, \vee, \wedge, F, T\rangle$ *fuzzy* $\langle [0, 1], \max, \min, 0, 1\rangle$, *bottleneck* $\langle \mathbb{R}^+ \cup \{+\infty\}, \max, \min, 0, \infty\rangle$, *probabilistic* $\langle [0, 1], \max, \hat{\times}, 0, 1\rangle$ (known as the Viterbi semiring), *weighted* $\langle \mathbb{R}^+ \cup \{+\infty\}, \min, \hat{+}, +\infty, 0\rangle$. Capped operators stand for their arithmetic equivalent.

2.2 Quantitative Controller Operator

A controlling strategy [12] is a run-time execution trace of a controller C that follows the behaviour of a target A. The resulting behaviour is denoted by $C \triangleright^{\mathbb{K}} A$, where $\mathbb{K}$ is the semiring used for specifying quantities on each executed action so that it is possible to quantitatively estimate the contribution of the countermeasures in the system workflow. Indeed, injecting a controller in a possible point of failure may increase, *e.g.*, the cost of the system, especially when it is activated to react to an attack.

Generalized Process Algebra. In this paper we model both a controller and a target as GPA processes [9]. GPA, i.e., *Generalized Process Algebra*, is a quantitative process algebra, whose transitions are labelled by pairs (a, k) where k is a quantity of a semiring associated to an action a.

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*

$$P ::= 0 \mid (a, k).P \mid P + P \mid P \|_L P \mid X$$

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

In order to give the flavour of the meaning of GPA operators, we informally describe their semantics:[2] process 0 describes inaction or termination; $(a, k).P$ performs action a with weight k, and then it evolves into P; $P + P'$ non-deterministically behaves as either P or P'; $P \|_L P'$ describes the process in which P and P' proceed concurrently and independently on all the actions that are not in L. On the other hand, all the actions in L are synchronisation points, meaning that the computation advances if and only if both P and P' perform the same action in L at the same time. $X \triangleq P$ allows to associate the behaviour of a process P (body) with a process variable name X (identifier).

Semantics Definitions for Quantitative Control-Rules. To denote controller and (its) target processes, hereafter we will use C and A respectively. The alphabets of C, A, and of the resulting process $C \triangleright^{\mathbb{K}} A$ are different. C may perform *control* actions of the form a, $\boxminus a$, $\boxplus a.b$ for $a, b \in Act$, denoting respectively the actions of *acceptance*, *suppression*, and *insertion*, which regulate the actions of A. The resulting process $C \triangleright^{\mathbb{K}} A$ may perform internal actions, denoted by τ, as a consequence of suppression. Each action of C, A, and $C \triangleright^{\mathbb{K}} A$ is associated with a value of a semiring $\mathbb{K}$, i.e., (a, k), where $k \in \mathbb{K}$ is a quantity associated with this action a.

Table 1. Semantics definitions for quantitative control-rules.

$$\frac{C \xrightarrow{a,k} C' \quad A \xrightarrow{a,k'} A'}{C \triangleright^{\mathbb{K}} A \xrightarrow{a,k \times k'} C' \triangleright^{\mathbb{K}} A'} \text{(A)} \qquad \frac{C \xrightarrow{\boxminus a,k} C' \quad A \xrightarrow{a,k'} A'}{C \triangleright^{\mathbb{K}} A \xrightarrow{\tau,k \times k'} C' \triangleright^{\mathbb{K}} A'} \text{(S)} \qquad \frac{C \xrightarrow{\boxplus a.b,k} C' \quad A \xrightarrow{a,k'} A'}{C \triangleright^{\mathbb{K}} A \xrightarrow{b,k} C' \triangleright^{\mathbb{K}} A} \text{(I)}$$

C follows the execution trace of A step by step, and it reacts to each step of the target according to one of the rules in Table 1. Note that, neither the controller nor the target performs τ actions independently.

The *acceptance rule* (A) in Table 1 constrains a controller and a target to perform the same action, in order for it to be observed in the resulting behaviour. In particular, if A performs an action a with a weight k', and the same action is performed by C with a weight k (so it is allowed on the system), then $E \triangleright^{\mathbb{K}} F$ performs a with an observed value that is the $\times$ of those of the controller and target, i.e., $k \times k'$.

The *suppression rule* allows C to hide an action a, but it counts its weight because it has been executed by A anyway. Hence, the suppression rule (S) in Table 1 allows the controller to hide target actions by performing a control action $\boxminus a$ with a measure k. The target wants to perform an action a with a weight k', but the action is not performed by the controlled entity and the observed result

[2] The interested reader can find the formal semantics of GPA processes defined in [9].

is a τ action, with the value computed as the product $k \times k'$ of the suppressing and the target actions. Then $C \triangleright^{\mathbb{K}} A$ performs an action τ that *suppresses* action a, i.e., a becomes invisible from external observation.

Finally, the *insertion rule* (I) in Table 1 describes the capability of correcting some undesirable behaviour of a target A: it inserts another action in A's execution trace by performing a control action $\boxplus a$ followed by an action b. The insertion cost corresponds to the value of C only, i.e., k; this accounts for the fact that A does not perform any action, but it rather stays in its current state.

2.3 Related Work

There is a significant bulk of work devoted to the enforcement of security mechanisms, *e.g.*, [18,23,29]. As foremost examples (due to similarities with respect to this work) we recall security automata [29], designed to prevent bad executions, and edit automata [4], which are able to edit their input sequences by suppressing, inserting, or replacing observed actions. One can also use concurrent languages (*e.g.*, process algebras), to model both the target and the control system in the same formalism [17,23,26].

As a prototypical example, we choose GPA process algebra [9], featuring a CSP-style synchronisation 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. 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 by Easwaran et al. in [16] in the context of software monitoring, taking into account rewards and penalties. In [27], the optimal policy can be derived by solving the optimisation problem of a Markov Decision Process. Bielova and Massacci propose in [5] a notion of *edit distance* among traces, which extends to an ordering over controllers. In [14], a notion of cost similar to the one we propose is used to compare several enforcement mechanisms that are correct (in the boolean sense). In this work, we follow some intuitive leads from [24] to move from qualitative to quantitative enforcement, and generalise that idea by using semirings. In [15] the possibility that the controller allows some policy violations is quantified over traces for non-safety policies, where a controller cannot be both correct and fully transparent. In [10], the authors use a notion of *lazy* controllers, which are able to check the security of a system at some point in time, proposing a probabilistic quantification of the expected risk. In the context of access control, Molloy et al. [28] use a machine-learning approach to predict a decision for a given request, and, at the same time, to balance the risk of error against the cost of contacting the real mechanism to get a decision. Non-binary measures of security have also been considered for access-control systems, *e.g.*, in [11,30].

In [13], Degano et al. propose a formal framework to specify and enforce quantitative security policies. The framework consists of *(i)* a stochastic process

calculus to express the measurable space of computations in terms of Continuous Time Markov Chains; *(ii)* a stochastic modal logic (a variant of CSL) to represent the bound constraints on execution speed; *(iii)* potential or actual enforcement mechanisms of quantitative security policies. The potential enforcement computes the probability of policy violations, thus supporting the user to accept/discard a component when the probability of security violation is below/above a suitable chosen threshold. The actual enforcement computes instead the deviation of execution speed from an acceptable rate.

In [7] the authors take advantage of an operational semantics with the aim to predict quantitative measures on systems describing cryptographic protocols. Moreover, they also introduce a possible attacker in the model. The transitions of the system carry enhanced labels: rates are assigned to transitions by only looking at these labels. Finally, transition systems are mapped to Markov chains and an evaluation of system performance is obtained by using standard tools.

In [2] the authors investigate usage automata, a formal model for specifying policies on the usage of resources. Usage automata extend finite state automata with some additional features, parameters and guards, that improve their expressivity. The authors check the decidability if a given computation complies with a usage policy.

3 Quantitative Security Approach

In the literature on qualitative enforcement of secure systems, several approaches have been developed, as we have recalled in Sect. 2.3. A possible approach for the specification, analysis, and synthesis of secure systems is based on the *open system* paradigm [22], where the considered system and a possible malicious agent interacting with it are represented as two processes that work in parallel.

The same approach can be used when we pass from a qualitative to a quantitative analysis of such system S. The unspecified part of S is a component whose behaviour is not known a priori, and we want S to be quantitatively secure, whatever the behaviour of such unspecified components is. A is the possible attacker whose behaviour is a priori unknown; $L \subseteq Act$ is the set of possible synchronisation actions; thus $S\|_L A$ is the overall (partially specified) system, on which we require that:

$$\forall A \quad S\|_L A \models_t \phi.$$

ϕ is a logic formula expressing some behavioural requirements, such as security requirements as well as performance or cost constraints (i.e., non-functional requirements), and t denotes a required level of satisfaction: the evaluation of ϕ with respect to $S\|_L A$ has to be equal to t. Formally,

Definition 3 ($\models_k$). *A process P satisfies a c-HM formula ϕ with a threshold-value t, i.e., $P \models_t \phi$, if and only if the interpretation of ϕ on P is equal to t. Formally: $P \models_t \phi \Leftrightarrow t = [\![\phi]\!]_P$.*

Even if it is not always possible to check all different behaviours of component A, nevertheless it is possible to define distinct countermeasures that follow the

rules of the controller operator $\triangleright^{\mathbb{K}}$ defined in Table 1. These countermeasures are specified as execution traces of a controller process denoted by C. They guarantee the system to properly work by forcing the desired behaviour of unspecified components, in such a way that the system satisfies ϕ according to a predefined value t. Hence, the question here is if there exists an implementation that, by monitoring the behaviour of an unspecified component A, guarantees S to satisfy the required security property with a certain value t:

$$\exists C \quad \forall A \quad S\|_L(C \triangleright^{\mathbb{K}} A) \models_t \phi$$

First of all we apply a QPMC function inspired from the work in [25], with the purpose to evaluate ϕ with respect to the behaviour of S. In this way we obtain a new formula $\phi' = \phi_{//_S}$ and we only have to monitor the attacker's behaviour A. ϕ' represents the necessary and sufficient conditions that $C \triangleright^{\mathbb{K}} A$ has to satisfy in order to guarantee the security of S. Indeed, the problem we have to solve reduces to the following one:

$$\exists C \quad \forall A \quad (C \triangleright^{\mathbb{K}} A) \models_t \phi' \tag{1}$$

It is worth noting that we neither know the behaviour of the controller process C, nor the one of attacker A. The only information we have is the semantics rules of the controller operator $\triangleright^{\mathbb{K}}$. Based on that, we have developed a quantitative partial model checking function able to refine the formula $\phi' \in$ c-HM into a binary formula $\phi'' \in$ c-HM^2 able to specify the set of quantitative controller execution traces for any attacker execution trace. The basic idea is that, knowing the possible reaction rules that drive the behaviour of a controller and the quantitative security requirements that S has to satisfy, it is possible to find the necessary and sufficient condition both the controller and the target has to quantitatively satisfy in order to assure the system security. Indeed, we consider both the controller and the target behaviours as unknown.

To this aim, we propose a variant of a quantitative Hennessy-Milner logic, the c-HM logic firstly proposed in [25]; thus, we can specify a property on couples of actions, extending it to c-HM^2 (see Sect. 3.1). Afterwards, we define a different version of the QPMC function [25], allowing us to refine a formula ϕ' with respect to the semantics definition of the controller operator (see Sect. 4). This refinement allows us to write each action of the execution trace of the controlled process as a couple of actions, respectively representing the weight contribution to that action of both the controller and the target.

3.1 Binary C-Semiring Hennessy-Milner Logic (c-HM^2)

We start by assembling the transition system on which c-HM^2 is defined:

Definition 4 (MLTS). *A (finite) Multi-Labelled Transition-System (MLTS) is a five-tuple MLTS* $= (S, Act^2, \mathbb{K}, T, s_0)$*, where* S *is the countable (finite) state-space,* $s_0 \in S$ *is the initial state,* Act^2 *is a finite set of transition labels, where each label is a couple of labels in* Act*, i.e., the label* $\langle a_1, a_2 \rangle \in Act^2$ *and* $a_1, a_2 \in L$*.*

$\mathbb{K}$ *is a semiring used for the definition of transition weights, and* $T : (S \times Act^2 \times S) \longrightarrow \mathbb{K}$ *is the transition weight-function.*

Definition 5 syntactically defines the correct formulas given over an MLTS.

Definition 5 (Syntax). *Given an MLTS* $M = \langle S, Act^2, \mathbb{K}, T, s_0 \rangle$*, and let* $\tilde{a} \in Act^2$*, the syntax of a formula* $\phi \in \Phi_M$ *is as follows, where* $k \in K$*:*

$$\phi ::= k \mid \phi_1 = \phi_2 \mid \phi_1 + \phi_2 \mid \phi_1 \times \phi_2 \mid \phi_1 \sqcap \phi_2 \mid \langle \tilde{a} \rangle \phi \mid [\tilde{a}]\phi$$

The semiring operators $+$, $\sqcap$ (the glb), and $\times$ are used in place of classical logic operators $\vee$ and $\wedge$, in order to compose the truth values of two formulas together. Truth values are all the $k \in K$. In particular, while *false* corresponds to $\mathbf{0}$, we can have different degrees of *true*, where "full truth" is $\mathbf{1}$. As a reminder, when the $\times$ operator is idempotent, then $\times$ and $\sqcap$ coincide (Sect. 2.1). Moreover we can use $=$ to compare the evaluation of two formulas: the result is $\mathbf{1}$ if they are both evaluated to the same $k \in K$, $\mathbf{0}$ otherwise (i.e., it corresponds to $\Leftrightarrow$ in boolean logic).[3] Finally, we have the two classical modal operators, i.e., "possibly" ($\langle \cdot \rangle$), and "necessarily" ($[\cdot]$).

Table 2. Semantics of c-HM^2. $\sum(\emptyset) = \mathbf{0}$ and $\sqcap(\emptyset) = \mathbf{1}$.

$$
\begin{array}{ll}
[\![k]\!](s) & = k \in K \ \ \forall s \in S \\
[\![\phi_1 = \phi_2]\!](s) & = \begin{cases} \mathbf{1} \ if \ [\![\phi_1]\!](s) = [\![\phi_2]\!](s) \\ \mathbf{0} \ otherwise \end{cases} \\
[\![\phi_1 + \phi_2]\!](s) & = [\![\phi_1]\!](s) + [\![\phi_2]\!](s) \\
[\![\phi_1 \times \phi_2]\!](s) & = [\![\phi_1]\!](s) \times [\![\phi_2]\!](s) \\
[\![\phi_1 \sqcap \phi_2]\!](s) & = [\![\phi_1]\!](s) \sqcap [\![\phi_2]\!](s) \\
[\![\langle a \rangle \phi]\!](s) & = \sum_{(s \xrightarrow{(a,k_a)} s') \in T} (k_a \times [\![\phi]\!](s')) \\
[\![[a]\phi]\!](s) & = \prod_{(s \xrightarrow{(a,k_a)} s') \in T} (k_a \times [\![\phi]\!](s'))
\end{array}
$$

The semantics of a formula ϕ is given on a finite MLTS $M = \langle S, Act^2, \mathbb{K}, T, s_0 \rangle$, where the set of states S corresponds to the set of finite GPA processes. The purpose is to check the specification defined by ϕ over the behaviour of a couple of GPA processes. The semantics of a formula, $[\![\,]\!]_M : (\Phi_M \times S) \longrightarrow K$ (see Table 2), computes a semiring value associated with a formula in a given state $s \in S$ of an MLTS M.

In Table 2 and in the following (when clear from the context), we omit M from $[\![\,]\!]_M$ for the sake of readability. It is worth noting that, due to the expressive

[3] We can think of further operators between formulas, *e.g.*, $\{=, \geq, \approx_\epsilon\}$.

power of c-HM^2, we deal with *safety properties*, *e.g.*, properties expressing that, if something goes wrong, then it can be detected in a finite number of steps.

Note that, the notion of satisfiability given in Definition 3 holds also for a c-HM^2 formula ϕ on a binary MLTS $M = \langle S, Act^2, \mathbb{K}, T, s_0 \rangle$. A binary MLTS may also represent a couple of processes composed as an *independent combination* of two processes P_1 and P_2, hereafter denoted as (P_1, P_2).

Definition 6 (Binary Process). *Let P_1 and P_2 be two GPA processes. A* binary process $(P_1, P_2) \in GPA \times GPA$ *is the juxtaposition of two processes and it is fully characterised by the following rule:*

$$\frac{P_1 \xrightarrow{(a,h)} P_1' \quad P_2 \xrightarrow{(b,s)} P_2'}{(P_1, P_2) \xrightarrow{((a,b),h \times s)} (P_1', P_2')}$$

Note that, the semantic interpretation of a binary process is given through a binary MLTS. In particular, according to the definition of binary transition function, the set of states of a binary process is the union of the two sets of states of both the processes. In this way, either both processes perform an action being in the same state or, they are *asynchronous* processes, i.e., both component P_1 and P_2 contribute in the transition of the combined process (P_1, P_2), even when one of the two performs the 0 action.

4 Quantitative Partial Model Checking for Controller Operator

In order to solve the problem in Eq. 1, we define a QPMC function with the purpose to evaluate $\phi' \in$ c-HM with respect to the application of controller rules, i.e., to controller strategies. As a remainder, ϕ' is obtained from the initial ϕ by removing S from the parallel computation, and "adding" it to ϕ ($\phi' = \phi_{//_S}$, see Sect. 3). Our goal in this section is to obtain a refinement of ϕ', i.e., $\phi'' \in$ c-HM^2, which also depends on $\triangleright^{\mathbb{K}}$.

The QPMC function we use to achieve such goal is defined in Table 3. Being the logic closed with respect to the QPMC function, the interpretation of the formulas obtained through the application of the function is straightforward. Theorem 1 proposes a result similar to the one in [1].

Theorem 1. *Let C and A two processes in GPA such that $C \triangleright^{\mathbb{K}} A \in GPA$ and (C, A) is a binary process in GPA $\times$ GPA, $\mathbb{K}$ a totally ordered c-semiring with $k \in K$, as well as ϕ' a c-HM formula and $\phi'' = \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi)$ a c-HM^2 formula, the following holds:*

$$[\![\phi']\!]_{(C \triangleright^{\mathbb{K}} A)} = [\![\phi'']\!]_{(C,A)}$$

Due to this result, the problem in Eq. 1 can be simplified as follows:

$$\exists C \; \forall A \quad (C, A) \models_t \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi') \tag{2}$$

where $\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi')$ is a c-HM^2 formula that describes, in a unique way for each action of the process A, that the reaction of a controller C guarantees the quantitative satisfaction of the initial system's requirements. Hence, it is worth observing that a security controller needs to make a *decision* in order to select the *best* reaction (if any). This decision is supported by the quantitative value associated to each reaction. However, in general, the decision for a single action can change according to the actions previously executed by A. For instance, in the Chinese Wall policy (see an example in Sect. 5), a user can *a priori* access to a resource from any company, unless in the past she has accessed to a resource from another company in the same conflict-of-interest class.

Moreover, dealing with quantitative aspects, it is important to distinguish between the decision process, i.e., C, and the actual implementation of the controlled process, i.e., $C \triangleright^{\mathbb{K}} A$. Indeed, it is possible to specify the best editing strategy by associating particular costs with actions. For instance, by setting the acceptance cost to a minimum, we showed that it is always the best strategy to accept a correct action. Similarly, by associating an infinite cost with the suppression of a particular action, we can model the concept of *uncontrollable action* [3], that is, an action that has to be accepted, such as the tick of a clock.

Table 3. A QPMC function (i.e., $\mathcal{W}$) for quantitative controller operator $\triangleright^{\mathbb{K}}$.

$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, k) = k$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1 = \phi_2) = (\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1) = \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_2))$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1 \times \phi_2) = \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1) \times \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_2)$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1 + \phi_2) = \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1) + \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_2)$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1 \sqcap \phi_2) = \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_1) \sqcap \mathcal{W}(C \triangleright^{\mathbb{K}} A, \phi_2)$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, [a]\phi) = \prod_{C \triangleright^{\mathbb{K}} A \xrightarrow{(a,k_a)} (C \triangleright^{\mathbb{K}} A)'} (k_a^{(a,a)} \times [(a,a)]\mathcal{W}(C' \triangleright^{\mathbb{K}} A', \phi)) \sqcap \sqcap (k_a^{(\boxplus b.a,b)} \times [(\boxplus b.a, b)]\mathcal{W}(C' \triangleright^{\mathbb{K}} A, \phi))$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, [\tau]\phi) = \prod_{C \triangleright^{\mathbb{K}} A \xrightarrow{(\tau,k_\tau)} (C \triangleright^{\mathbb{K}} A)'} k_\tau^{(\boxminus a,a)} \times ([(\boxminus a, a)]\mathcal{W}(C' \triangleright^{\mathbb{K}} A', \phi))$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \langle a\rangle\phi) = \sum_{C \triangleright^{\mathbb{K}} A \xrightarrow{(a,k_a)} (C \triangleright^{\mathbb{K}} A)'} (k_a^{(a,a)} \times (\langle(a,a)\rangle\mathcal{W}(C' \triangleright^{\mathbb{K}} A', \phi)) + + (k_a^{(\boxplus b.a,b)} \times \langle(\boxplus b.a, b)\rangle\mathcal{W}(C' \triangleright^{\mathbb{K}} A, \phi))$$
$$\mathcal{W}(C \triangleright^{\mathbb{K}} A, \langle\tau\rangle\phi) = \sum_{C \triangleright^{\mathbb{K}} A \xrightarrow{(\tau,k_\tau)} (C \triangleright^{\mathbb{K}} A)'} (k_\tau^{(\boxminus a,a)} \times \langle(\boxminus a, a)\rangle\mathcal{W}(C' \triangleright^{\mathbb{K}} A', \phi))$$

5 A Simple Example

To exemplify our approach, let us consider a very simple example. For sake of simplicity, we omit the system S: the goal is to show how the QPMC function works with respect to controller operations. To do this, let us now consider a well-known access-control policy for distributed systems: the *Chinese Wall* policy.

To evaluate the security level of $C \triangleright^{\mathbb{K}} A$, each action is weighted with a semiring value expressing a security-evaluation score for that action. Security, trust, functionality, and performance can be represented by different semirings. In this section, we use weights from $\mathbb{S} = \langle \{\underline{i}, \underline{l}, \underline{m}, \underline{g}, \underline{e}\}, \max, \min, \underline{i}, \underline{e} \rangle$, where the chain $\underline{i}nsecure \leq \underline{l}ow \leq \underline{m}edium \leq \underline{g}ood \leq \underline{e}xcellent$ models a set of security levels. When we compose two levels together we choose the worst, while preference goes to the higher level.

Given two sets of resources (*e.g.*, files or data) V and W, such policy states that one can choose to access either to V or to W, but if an access to V is performed (setting the security level of access to V to $\underline{e}$) then it is no more possible to access to W; consequently, the access to W has security level $\underline{i}$. Clearly, this also holds vice-versa, i.e., if we open an element x of W then we can not access to any element in V. Note that, in this example, the required security level to access to set W, i.e., $\underline{l}$, is less than the required security level to access to V, that is $\underline{e}$. The reason is that V collects more sensitive information. The Chinese Wall policy is expressed by a formula $\phi = \phi_1 + \phi_2$ where

$$\phi_1 = [access\_V]\underline{e} \times [access\_W]\underline{i} \qquad \phi_2 = [access\_W]\underline{l} \times [access\_V]\underline{i}.$$

In this example, we consider an insertion controller $\triangleright^{\mathbb{S}}$, and we require $C \triangleright^{\mathbb{S}} A$ to satisfy the Chinese Wall policy with a security equal to $\underline{g}$.

By using the QPMC function with respect to the controller operator $\triangleright^{\mathbb{S}}$, we have that:

$$C \triangleright^{\mathbb{S}} A \models_{\underline{g}} \phi \Leftrightarrow (C, A) \models_{\underline{g}} \phi'$$

where $\phi' = \mathcal{W}(C \triangleright^{\mathbb{S}} A, \phi) = \phi'_1 + \phi'_2$, and

$$\begin{aligned}
\phi'_1 = \mathcal{W}(C \triangleright^{\mathbb{S}} A, \phi_1) = {} & (k_{access_V}^{(access_V, access_V)} \times [(access_V, access_V)]\underline{e}) \\
& \sqcap (k_{access_V}^{(\boxplus access_W.access_V, access_W)} \\
& \times [(\boxplus access_W.access_V, access_W)]\underline{e}) \\
& \times (k_{access_W}^{(access_W, access_W)} \times [(access_W, access_W)]\underline{i}) \\
& \sqcap (k_{access_W}^{(\boxplus access_V.access_W, access_V)} \\
& \times [(\boxplus access_V.access_W, access_V)]\underline{i}) \\
\phi'_2 = \mathcal{W}(C \triangleright^{\mathbb{S}} A, \phi_2) = {} & (k_{access_W}^{(access_W, access_W)} \times [(access_W, access_W)]\underline{l}) \\
& \sqcap (k_{access_W}^{(\boxplus access_V.access_W, access_V)} \\
& \times [(\boxplus access_V.access_W, access_V)]\underline{l}) \\
& \times (k_{access_V}^{(access_V, access_V)} \times [(access_V, access_V)]\underline{i}) \\
& \sqcap (k_{access_V}^{(\boxplus access_W.access_V, access_W)} \\
& \times [(\boxplus access_W.access_V, access_W)]\underline{i})
\end{aligned}$$

According to Theorem 1, to verify if $C \triangleright^{\mathbb{S}} A$ quantitatively satisfies the Chinese Wall policy with a security level $\underline{g}$, it is necessary and sufficient to evaluate ϕ' with respect the binary process $(\overline{C}, A)$. A priori we do not know the behaviour of A, however, due to the quantitative nature of the proposed framework, we can infer some constraints on the controller process C, which help the synthesis of the *best* controller, if it exists.

As a remainder, the weight of an accepted action is equal to the product (i.e., $\times$) of the weights of both the actions respectively performed by C and A, while the weight of an inserted action is equal only to the weight associated with the action of C. This leads to the following considerations:

- If the attacker does not perform the correct action, *e.g.*, it tries to access to W after accessing to V (or vice versa), the controller C may insert the correct action $access\_V$ with an appropriate security level, *e.g.*, better than $\underline{g}$. In this way, the controller assures that the Chinese Wall policy is satisfied with the required security level $\underline{g}$.
- If the attacker performs the correct action, but with a security level worse than the required one, *e.g.*, $\underline{g}$ in the example, the controller, accepting the correct action, does not increase the level of security. Thus, the Chinese Wall policy is not satisfied because the required security level is not respected. This is the case in which both C and A perform a valid sequence of actions, *e.g.*, one $access\_V$ each, but the level of one of these actions is worse than $\underline{g}$. In this case, the controller guarantees that the Chinese Wall Policy is not violated by not changing the security level of the attacker actions, and accepting the correct action. However, also in this case as well as in the previous one, C can insert the correct action with the correct security level in such a way to not halt the execution and, at the same time, guarantee the satisfaction of ϕ. It is worth noting that, another possible scenario may happen when an agent A try to access to W with a security level l. This does not violate the requirement imposed by ϕ, but it violates the satisfaction requirements, because $\underline{l}$ is worse than $\underline{g}$. Also in this case, C can fix the execution trace by inserting the correct action with a more appropriate security level.

6 Conclusion

We have presented a verification framework to study quantitative properties associated with a formula ϕ, i.e., properties with an associated weight. Such a value is interpreted as how costly the verification of a property is. The conundrum has consisted in investigating controller-agents C accepting, suppressing, or inserting actions in the behaviour of an attacker A, while considering the correct functioning of a system S. The question we have address in this paper is $\exists C\ \forall A\ S \| (C \triangleright^{\mathbb{K}} A) \models_t \phi$. As in Sect. 1, we again come across a triangled structure: the drama triangle[4] is a psychological and social model of human

[4] First described by Stephen Karpman, M.D., in his 1968 article *"Fairy Tales and Script Drama Analysis"*.

interaction used in psychology and psychotherapy. At its vertices we find the Victim (S), the Persecutor (A), and the Rescuer (C). With the aid of a QPMC function we remove S from the global parallel computation (moving it into ϕ), and we refine ϕ' investigating the duties of C and A in order to satisfy ϕ: such approach helps us to better understand C and A separately.

In the future we plan to extend this work in several ways. For instance, we plan to have a multidimensional decomposition of properties, instead of a bi-dimensional one as in this paper: we would like to follow the pioneering proposal in [20], thus decomposing quantitative properties satisfied by an n-ary context into n local quantitative constraints, each of them satisfied by a unary (quantitative) context. Each context represents a different component of a distributed system. In such a way, we can improve the approach by taking into account fully-distributed systems with multiple components and attackers. Another direction is the extension of the framework to use more than one measure in order to evaluate a context. Such measures can be combined and ordered, *e.g.*, by using the lexicographical ordering, in such a way that controlling strategies can be selected with respect to the optimisation of the trade-off between some of them. Finally, we would like to manage infinite contexts by extending our logic to deal with fix-points; to achieve this goal, suggestions could come from the work in [21].

References

1. Andersen, H.R.: Partial model checking. In: LICS 1995, p. 398. IEEE Computer Society (1995)
2. Bartoletti, M., Degano, P., Ferrari, G.L., Zunino, R.: Model checking usage policies. Math. Struct. Comput. Sci. **25**(3), 710–763 (2015)
3. Klaedtke, F., Zălinescu, E., Jugé, V., Basin, D.: Enforceable security policies revisited. In: Degano, P., Guttman, J.D. (eds.) Principles of Security and Trust. LNCS, vol. 7215, pp. 309–328. Springer, Heidelberg (2012)
4. 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)
5. Bielova, N., Massacci, F.: Predictability of enforcement. In: Erlingsson, Ú., Zannone, N., Wieringa, R. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 73–86. Springer, Heidelberg (2011)
6. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44**(2), 201–236 (1997)
7. Bodei, C., Curti, M., Degano, P., Priami, C.: A quantitative study of two attacks. Electr. Notes Theor. Comput. Sci. **121**, 65–85 (2005)
8. McQueen, M., Boyer, W.: Ideal based cyber security technical metrics for control systems. In: Hämmerli, B.M., Lopez, J. (eds.) CRITIS 2007. LNCS, vol. 5141, pp. 246–260. Springer, Heidelberg (2008)
9. Buchholz, P., Kemper, P.: Quantifying the dynamic behavior of process algebras. In: Gilmore, S., de Luca, L. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, p. 184. Springer, Heidelberg (2001)
10. Caravagna, G., Costa, G., Pardini, G.: Lazy security controllers. In: Samarati, P., Petrocchi, M., Jøsang, A. (eds.) STM 2012. LNCS, vol. 7783, pp. 33–48. Springer, Heidelberg (2013)

11. Cheng, P.C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Reninger, A.S.: Fuzzy multi-level security: an experiment on quantified risk-adaptive access control. In: Proceedings of the 2007 IEEE S&P, pp. 222–230. IEEE Computer Society (2007)
12. Ciancia, V., Martinelli, F., Ilaria, M., Morisset, C.: Quantitative evaluation of enforcement strategies (position paper). In: Danger, J.-L., Debbabi, M., Marion, J.-Y., Garcia-Alfaro, J., Heywood, N.Z. (eds.) FPS 2013. LNCS, vol. 8352, pp. 178–186. Springer, Heidelberg (2014)
13. Degano, P., Mezzetti, G., Ferrari, G.-L.: On quantitative security policies. In: Malyshkin, V. (ed.) PaCT 2011. LNCS, vol. 6873, pp. 23–39. Springer, Heidelberg (2011)
14. 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)
15. Drábik, P., Martinelli, F., Morisset, C.: A quantitative approach for inexact enforcement of security policies. In: Freiling, F.C., Gollmann, D. (eds.) ISC 2012. LNCS, vol. 7483, pp. 306–321. Springer, Heidelberg (2012)
16. Easwaran, A., Kannan, S., Lee, I.: Optimal control of software ensuring safety and functionality. Tech. Rep. MS-CIS-05-20, University of Pennsylvania (2005)
17. 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)
18. Khoury, R., Tawbi, N.: Which security policies are enforceable by runtime monitors? a survey. Computer Science Review **6**(1), 27–45 (2012)
19. Köpf, B., Malacaria, P., Palamidessi, C.: Quantitative security analysis (Dagstuhl seminar 12481). Dagstuhl Reports **2**(11), 135–154 (2013)
20. Larsen, K.G., Xinxin, L.: Compositionality through an operational semantics of contexts. J. Logic Comput. **1**(6), 761–795 (1991)
21. Lluch-Lafuente, A., Montanari, U.: Quantitative mu-calculus and CTL defined over constraint semirings. TCS **346**(1), 135–160 (2005)
22. Martinelli, F.: Analysis of security protocols as open systems. TCS **290**(1), 1057–1106 (2003)
23. Martinelli, F., Matteucci, I.: Through modeling to synthesis of security automata. ENTCS **179**, 31–46 (2007)
24. 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)
25. Martinelli, F., Matteucci, I., Santini, F.: Quantitative security on distributed systems. In: EPTCS (ed.) Proceedings of the 13th International Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL 2015) (2015) (accepted for publication)
26. Martinelli, F., Matteucci, I.: Partial model checking, process algebra operators and satisfiability procedures for (automatically) enforcing security properties. Tech. rep, IIT-CNR (2005)
27. Martinelli, F., Morisset, C.: Quantitative access control with partially-observable markov decision processes. In: Proceedings of CODASPY 2012, pp. 169–180. ACM (2012)
28. Molloy, I., Dickens, L., Morisset, C., Cheng, P.C., Lobo, J., Russo, A.: Risk-based security decisions under uncertainty. In: Proceedings of the second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, pp. 157–168. ACM (2012)

29. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. **3**(1), 30–50 (2000)
30. Zhang, L., Brodsky, A., Jajodia, S.: Toward Information Sharing: Benefit And Risk Access Control (BARAC). In: Proceedings of POLICY 2006, pp. 45–53 (2006)