Measuring Masking Fault-Tolerance

Pablo F. Castro[1,3]([✉]) [iD], Pedro R. D'Argenio[2,3,4] [iD],
Ramiro Demasi[2,3] [iD], and Luciano Putruele[1,3]

[1] Departamento de Computación, FCEFQyN,
Universidad Nacional de Río Cuarto, Río Cuarto,
Córdoba, Argentina
{pcastro,lputruele}@dc.exa.unrc.edu.ar
[2] FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
{dargenio,rdemasi}@famaf.unc.edu.ar
[3] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET),
Buenos Aires, Argentina
[4] Saarland University, Saarbrücken, Germany

Abstract. In this paper we introduce a notion of fault-tolerance distance between labeled transition systems. Intuitively, this notion of distance measures the degree of fault-tolerance exhibited by a candidate system. In practice, there are different kinds of fault-tolerance, here we restrict ourselves to the analysis of masking fault-tolerance because it is often a highly desirable goal for critical systems. Roughly speaking, a system is masking fault-tolerant when it is able to completely mask the faults, not allowing these faults to have any observable consequences for the users. We capture masking fault-tolerance via a simulation relation, which is accompanied by a corresponding game characterization. We enrich the resulting games with quantitative objectives to define the notion of masking fault-tolerance distance. Furthermore, we investigate the basic properties of this notion of masking distance, and we prove that it is a directed semimetric. We have implemented our approach in a prototype tool that automatically computes the masking distance between a nominal system and a fault-tolerant version of it. We have used this tool to measure the masking tolerance of multiple instances of several case studies.

1 Introduction

Fault-tolerance allows for the construction of systems that are able to overcome the occurrence of faults during their execution. Examples of fault-tolerant systems can be found everywhere: communication protocols, hardware circuits, avionic systems, cryptographic currencies, etc. So, the increasing relevance of critical software in everyday life has led to a renewed interest in the automatic

verification of fault-tolerant properties. However, one of the main difficulties when reasoning about these kinds of properties is given by their quantitative nature, which is true even for non-probabilistic systems. A simple example is given by the introduction of redundancy in critical systems. This is, by far, one of the most used techniques in fault-tolerance. In practice, it is well-known that adding more redundancy to a system increases its reliability. Measuring this increment is a central issue for evaluating fault-tolerant software, protocols, etc. On the other hand, the formal characterization of fault-tolerant properties could be an involving task, usually these properties are encoded using *ad-hoc* mechanisms as part of a general design.

The usual flow for the design and verification of fault-tolerant systems consists in defining a nominal model (i.e., the "fault-free" or "ideal" program) and afterwards extending it with faulty behaviors that deviate from the normal behavior prescribed by the nominal model. This extended model represents the way in which the system operates under the occurrence of faults. There are different ways of extending the nominal model, the typical approach is *fault injection* [20,21], that is, the automatic introduction of faults into the model. An important property that any extended model has to satisfy is the preservation of the normal behavior under the absence of faults. In [11], we proposed an alternative formal approach for dealing with the analysis of fault-tolerance. This approach allows for a fully automated analysis and appropriately distinguishes faulty behaviors from normal ones. Moreover, this framework is amenable to fault-injection. In that work, three notions of simulation relations are defined to characterize *masking*, *nonmasking*, and *failsafe* fault-tolerance, as originally defined in [15].

During the last decade, significant progress has been made towards defining suitable metrics or distances for diverse types of quantitative models including real-time systems [19], probabilistic models [12], and metrics for linear and branching systems [6,8,18,23,29]. Some authors have already pointed out that these metrics can be useful to reason about the robustness of a system, a notion related to fault-tolerance. Particularly, in [6] the traditional notion of simulation relation is generalized and three different simulation distances between systems are introduced, namely *correctness*, *coverage*, and *robustness*. These are defined using quantitative games with *discounted-sum* and *mean-payoff* objectives.

In this paper we introduce a notion of fault-tolerance distance between labelled transition systems. Intuitively, this distance measures the degree of fault-tolerance exhibited by a candidate system. As it was mentioned above, there exist different levels of fault-tolerance, we restrict ourselves to the analysis of *masking fault-tolerance* because it is often classified as the most benign kind of fault-tolerance and it is a highly desirable property for critical systems. Roughly speaking, a system is masking fault-tolerant when it is able to completely mask the faults, not allowing these faults to have any observable consequences for the users. Formally, the system must preserve both the safety and liveness properties of the nominal model [15]. In contrast to the robustness distance defined in [6], which measures how many unexpected errors are tolerated by the implementation, we consider a

specific collection of faults given in the implementation and measure how many faults are tolerated by the implementation in such a way that they can be masked by the states. We also require that the normal behavior of the specification has to be preserved by the implementation when no faults are present. In this case, we have a bisimulation between the specification and the non-faulty behavior of the implementation. Otherwise, the distance is 1. That is, $\delta_m(N, I) = 1$ if and only if the nominal model N and $I \backslash F$ are not bisimilar, where $I \backslash F$ behaves like the implementation I where all actions in F are forbidden ($\backslash$ is Milner's restriction operator). Thus, we effectively distinguish between the nominal model and its fault-tolerant version and the set of faults taken into account.

In order to measure the degree of masking fault-tolerance of a given system, we start characterizing masking fault-tolerance via simulation relations between two systems as defined in [11]. The first one acting as a specification of the intended behavior (i.e., nominal model) and the second one as the fault-tolerant implementation (i.e., the extended model with faulty behavior). The existence of a masking relation implies that the implementation masks the faults. Afterwards, we introduce a game characterization of masking simulation and we enrich the resulting games with quantitative objectives to define the notion of *masking fault-tolerance distance*, where the possible values of the game belong to the interval $[0, 1]$. The fault-tolerant implementation is masking fault-tolerant if the value of the game is 0. Furthermore, the bigger the number, the farther the masking distance between the fault-tolerant implementation and the specification. Accordingly, a bigger distance remarkably decreases fault-tolerance. Thus, for a given nominal model N and two different fault-tolerant implementations I_1 and I_2, our distance ensures that $\delta_m(N, I_1) < \delta_m(N, I_2)$ whenever I_1 tolerates more faults than I_2. We also provide a weak version of masking simulation, which makes it possible to deal with complex systems composed of several interacting components. We prove that masking distance is a directed semimetric, that is, it satisfies two basic properties of any distance, reflexivity and the triangle inequality.

Finally, we have implemented our approach in a tool that takes as input a nominal model and its fault-tolerant implementation and automatically compute the masking distance between them. We have used this tool to measure the masking tolerance of multiple instances of several case studies such as a redundant cell memory, a variation of the dining philosophers problem, the bounded retransmission protocol, N-Modular-Redundancy, and the Byzantine generals problem. These are typical examples of fault-tolerant systems.

The remainder of the paper is structured as follows. In Sect. 2, we introduce preliminaries notions used throughout this paper. We present in Sect. 3 the formal definition of masking distance build on quantitative simulation games and we also prove its basic properties. We describe in Sect. 4 the experimental evaluation on some well-known case studies. In Sect. 5 we discuss the related work. Finally, we discuss in Sect. 6 some conclusions and directions for further work. Full details and proofs can be found in [5].

2   Preliminaries

Let us introduce some basic definitions and results on game theory that will be necessary across the paper, the interested reader is referred to [2].

A *transition system* (TS) is a tuple $A = \langle S, \Sigma, E, s_0 \rangle$, where S is a finite set of states, Σ is a finite alphabet, $E \subseteq S \times \Sigma \times S$ is a set of labelled transitions, and s_0 is the initial state. In the following we use $s \xrightarrow{e} s' \in E$ to denote $(s, e, s') \in E$. Let $|S|$ and $|E|$ denote the number of states and edges, respectively. We define $post(s) = \{ s' \in S \mid s \xrightarrow{e} s' \in E \}$ as the set of successors of s. Similarly, $pre(s') = \{ s \in S \mid s \xrightarrow{e} s' \in E \}$ as the set of predecessors of s'. Moreover, $post^*(s)$ denotes the states which are reachable from s. Without loss of generality, we require that every state s has a successor, i.e., $\forall s \in S : post(s) \neq \emptyset$. A run in a transition system A is an infinite path $\rho = \rho_0 \sigma_0 \rho_1 \sigma_1 \rho_2 \sigma_2 \cdots \in (S \cdot \Sigma)^w$ where $\rho_0 = s_0$ and for all i, $\rho_i \xrightarrow{\sigma_i} \rho_{i+1} \in E$. From now on, given a tuple $(x_0, \ldots, x_n)$, we denote x_i by $\mathrm{pr}_i((x_0, \ldots, x_n))$.

A *game graph* G is a tuple $G = \langle S, S_1, S_2, \Sigma, E, s_0 \rangle$ where S, Σ, E and s_0 are as in transition systems and (S_1, S_2) is a partition of S. The choice of the next state is made by Player 1 (Player 2) when the current state is in S_1 (respectively, S_2). A weighted game graph is a game graph along with a weight function v^G from E to $\mathbb{Q}$. A run in the game graph G is called a *play*. The set of all plays is denoted by Ω.

Given a game graph G, a *strategy* for Player 1 is a function $\pi : (S \cdot \Sigma)^* S_1 \to \Sigma \times S$ such that for all $\rho_0 \sigma_0 \rho_1 \sigma_1 \ldots \rho_i \in (S \cdot \Sigma)^* S_1$, we have that if $\pi(\rho_0 \sigma_0 \rho_1 \sigma_1 \ldots \rho_i) = (\sigma, \rho)$, then $\rho_i \xrightarrow{\sigma} \rho \in E$. A strategy for Player 2 is defined in a similar way. The set of all strategies for Player p is denoted by Π_p. A strategy for player p is said to be memoryless (or positional) if it can be defined by a mapping $f : S_p \to E$ such that for all $s \in S_p$ we have that $\mathrm{pr}_0(f(s)) = s$, that is, these strategies do not need memory of the past history. Furthermore, a play $\rho_0 \sigma_0 \rho_1 \sigma_1 \rho_2 \sigma_2 \ldots$ conforms to a player p strategy π if $\forall i \geq 0 : (\rho_i \in S_p) \Rightarrow (\sigma_i, \rho_{i+1}) = \pi(\rho_0 \sigma_0 \rho_1 \sigma_1 \ldots \rho_i)$. The *outcome* of a Player 1 strategy π_1 and a Player 2 strategy π_2 is the unique play, named $out(\pi_1, \pi_2)$, that conforms to both π_1 and π_2.

A *game* is made of a game graph and a boolean or quantitative objective. A *boolean objective* is a function $\Phi : \Omega \to \{0, 1\}$ and the goal of Player 1 in a game with objective Φ is to select a strategy so that the outcome maps to 1, independently what Player 2 does. On the contrary, the goal of Player 2 is to ensure that the outcome maps to 0. Given a boolean objective Φ, a play ρ is *winning* for Player 1 (resp. Player 2) if $\Phi(\rho) = 1$ (resp. $\Phi(\rho) = 0$). A strategy π is a *winning strategy* for Player p if every play conforming to π is winning for Player p. We say that a game with boolean objective is *determined* if some player has a winning strategy, and we say that it is memoryless determined if that winning strategy is memoryless. Reachability games are those games whose objective functions are defined as $\Phi(\rho_0 \sigma_0 \rho_1 \sigma_1 \rho_2 \sigma_2 \ldots) = (\exists i : \rho_i \in V)$ for some set $V \subseteq S$, a standard result is that reachability games are memoryless determined.

A *quantitative objective* is given by a *payoff* function $f : \Omega \to \mathbb{R}$ and the goal of Player 1 is to maximize the value f of the play, whereas the goal of Player 2 is to minimize it. For a quantitative objective f, the value of the game for a Player 1 strategy π_1, denoted by $v_1(\pi_1)$, is defined as the infimum over all the values resulting from Player 2 strategies, i.e., $v_1(\pi_1) = \inf_{\pi_2 \in \Pi_2} f(out(\pi_1, \pi_2))$. The value of the game for Player 1 is defined as the supremum of the values of all Player 1 strategies, i.e., $\sup_{\pi_1 \in \Pi_1} v_1(\pi_1)$. Analogously, the value of the game for a Player 2 strategy π_2 and the value of the game for Player 2 are defined as $v_2(\pi_2) = \sup_{\pi_1 \in \Pi_1} f(out(\pi_1, \pi_2))$ and $\inf_{\pi_2 \in \Pi_2} v_2(\pi_2)$, respectively. We say that a game is determined if both values are equal, that is: $\sup_{\pi_1 \in \Pi_1} v_1(\pi_1) = \inf_{\pi_2 \in \Pi_2} v_2(\pi_2)$. In this case we denote by $val(\mathcal{G})$ the value of game $\mathcal{G}$. The following result from [24] characterizes a large set of determined games.

Theorem 1. *Any game with a quantitative function f that is bounded and Borel measurable is determined.*

3   Masking Distance

We start by defining masking simulation. In [11], we have defined a state-based simulation for masking fault-tolerance, here we recast this definition using labelled transition systems. First, let us introduce some concepts needed for defining masking fault-tolerance. For any vocabulary Σ, and set of labels $\mathcal{F} = \{F_0, \dots, F_n\}$ not belonging to Σ, we consider $\Sigma_{\mathcal{F}} = \Sigma \cup \mathcal{F}$, where $\mathcal{F} \cap \Sigma = \emptyset$. Intuitively, the elements of $\mathcal{F}$ indicate the occurrence of a fault in a faulty implementation. Furthermore, sometimes it will be useful to consider the set $\Sigma^i = \{e^i \mid e \in \Sigma\}$, containing the elements of Σ indexed with superscript i. Moreover, for any vocabulary Σ we consider $\Sigma_M = \Sigma \cup \{M\}$, where $M \notin \Sigma$, intuitively, this label is used to identify masking transitions.

Given a transition system $A = \langle S, \Sigma, E, s_0 \rangle$ over a vocabulary Σ, we denote $A^M = \langle S, \Sigma_M, E^M, s_0 \rangle$ where $E^M = E \cup \{s \xrightarrow{M} s \mid s \in S\}$.

3.1   Strong Masking Simulation

Definition 1. *Let $A = \langle S, \Sigma, E, s_0 \rangle$ and $A' = \langle S', \Sigma_{\mathcal{F}}, E', s_0' \rangle$ be two transition systems. A' is* strong masking fault-tolerant *with respect to A if there exists a relation $\mathbf{M} \subseteq S \times S'$ between A^M and A' such that:*

(A) $s_0 \mathbf{M} s_0'$, and
(B) for all $s \in S, s' \in S'$ with $s \mathbf{M} s'$ and all $e \in \Sigma$ the following holds:
  (1) if $(s \xrightarrow{e} t) \in E$ then $\exists t' \in S' : (s' \xrightarrow{e} t' \wedge t \mathbf{M} t')$;
  (2) if $(s' \xrightarrow{e} t') \in E'$ then $\exists t \in S : (s \xrightarrow{e} t \wedge t \mathbf{M} t')$;
  (3) if $(s' \xrightarrow{F} t')$ for some $F \in \mathcal{F}$ then $\exists t \in S : (s \xrightarrow{M} t \wedge t \mathbf{M} t')$.

If such relation exists we say that A' is a strong masking fault-tolerant implementation *of A, denoted by $A \preceq_m A'$.*

We say that state s' is masking fault-tolerant for s when $s \; \mathbf{M} \; s'$. Intuitively, the definition states that, starting in s', faults can be masked in such a way that the behavior exhibited is the same as that observed when starting from s and executing transitions without faults. In other words, a masking relation ensures that every faulty behavior in the implementation can be simulated by the specification. Let us explain in more detail the above definition. First, note that conditions A, $B.1$, and $B.2$ imply that we have a bisimulation when A and A' do not exhibit faulty behavior. Particularly, condition $B.1$ says that the normal execution of A can be simulated by an execution of A'. On the other hand, condition $B.2$ says that the implementation does not add normal (non-faulty) behavior. Finally, condition $B.3$ states that every outgoing faulty transition (F) from s' must be matched to an outgoing masking transition (M) from s.

3.2   Weak Masking Simulation

For analysing nontrivial systems a weak version of masking simulation relation is needed, the main idea is that a weak masking simulation abstracts away from internal behaviour, which is modeled by a special action τ. Note that internal transitions are common in fault-tolerance: the actions performed as part of a fault-tolerant procedure in a component are usually not observable by the rest of the system.

The *weak transition relations* $\Rightarrow \; \subseteq S \times (\Sigma \cup \{\tau\} \cup \{M\} \cup \mathcal{F}) \times S$, also denoted as E_W, considers the *silent* step τ and is defined as follows:

$$\overset{e}{\Rightarrow} = \begin{cases} (\overset{\tau}{\rightarrow})^* \circ \overset{e}{\rightarrow} \circ (\overset{\tau}{\rightarrow})^* & \text{if } e \in \Sigma, \\ (\overset{e}{\rightarrow})^* & \text{if } e = \tau, \\ \overset{e}{\rightarrow} & \text{if } e \in \{M\} \cup \mathcal{F}. \end{cases}$$

The symbol $\circ$ stands for composition of binary relations and $(\overset{\tau}{\rightarrow})^*$ is the reflexive and transitive closure of the binary relation $\overset{\tau}{\rightarrow}$.

Intuitively, if $e \notin \{\tau, M\} \cup \mathcal{F}$, then $s \overset{e}{\Rightarrow} s'$ means that there is a sequence of zero or more τ transitions starting in s, followed by one transition labelled by e, followed again by zero or more τ transitions eventually reaching s'. $s \overset{\tau}{\Rightarrow} s'$ states that s can transition to s' via zero or more τ transitions. In particular, $s \overset{\tau}{\Rightarrow} s$ for every s. For the case in which $e \in \{M\} \cup \mathcal{F}$, $s \overset{e}{\Rightarrow} s'$ is equivalent to $s \overset{e}{\rightarrow} s'$ and hence no τ step is allowed before or after the e transition.

Definition 2. *Let $A = \langle S, \Sigma, E, s_0 \rangle$ and $A' = \langle S', \Sigma_{\mathcal{F}}, E', s_0' \rangle$ be two transition systems with Σ possibly containing τ. A' is* weak masking fault-tolerant *with respect to A if there is a relation $\mathbf{M} \subseteq S \times S'$ between A^M and A' such that:*

(A) $s_0 \; \mathbf{M} \; s_0'$
(B) for all $s \in S, s' \in S'$ with $s \; \mathbf{M} \; s'$ and all $e \in \Sigma \cup \{\tau\}$ the following holds:
(1) if $(s \overset{e}{\rightarrow} t) \in E$ then $\exists t' \in S' : (s' \overset{e}{\Rightarrow} t' \in E_W' \wedge t \; \mathbf{M} \; t')$;
(2) if $(s' \overset{e}{\rightarrow} t') \in E'$ then $\exists t \in S : (s \overset{e}{\Rightarrow} t \in E_W \wedge t \; \mathbf{M} \; t')$;
(3) if $(s' \overset{F}{\rightarrow} t') \in E'$ for some $F \in \mathcal{F}$ then $\exists t \in S : (s \overset{M}{\rightarrow} t \in E \wedge t \; \mathbf{M} \; t')$.

If such relation exists, we say that A' is a weak masking fault-tolerant imple-
mentation *of A, denoted by $A \preceq_m^w A'$.*

The following theorem makes a strong connection between strong and weak
masking simulation. It states that weak masking simulation becomes strong
masking simulation whenever transition $\rightarrow$ is replaced by $\Rightarrow$ in the original
automata.

Theorem 2. *Let $A = \langle S, \Sigma, E, s_0 \rangle$ and $A' = \langle S', \Sigma_{\mathcal{F}}, E', s_0' \rangle$. $\mathbf{M} \subseteq S \times S'$
between A^M and A' is a weak masking simulation if and only if:*

(A) $s_0 \mathbf{M} s_0'$, and
(B) for all $s \in S, s' \in S'$ with $s \mathbf{M} s'$ and all $e \in \Sigma \cup \{\tau\}$ the following holds:
  (1) if $(s \overset{e}{\Rightarrow} t) \in E_W$ then $\exists t' \in S' : (s' \overset{e}{\Rightarrow} t' \in E_W' \wedge t \mathbf{M} t')$;
  (2) if $(s' \overset{e}{\Rightarrow} t') \in E_W'$ then $\exists t \in S : (s \overset{e}{\Rightarrow} t \in E_W \wedge t \mathbf{M} t')$;
  (3) if $(s' \overset{F}{\Rightarrow} t') \in E_W'$ for some $F \in \mathcal{F}$ then $\exists t \in S : (s \overset{M}{\Rightarrow} t \in E_W \wedge t \mathbf{M} t')$

The proof of this is straightforward following the same ideas of Milner in [25].

A natural way to check weak bisimilarity is to *saturate* the transition system
[14, 25] and then check strong bisimilarity on the saturated transition system.
Similarly, Theorem 2 allows us to compute weak masking simulation by reduc-
ing this problem to compute strong masking simulation. Note that $\overset{e}{\Rightarrow}$ can be
alternatively defined by:

$$\frac{p \overset{e}{\rightarrow} q}{p \overset{e}{\Rightarrow} q} \qquad \frac{}{p \overset{\tau}{\Rightarrow} p} \qquad \frac{p \overset{\tau}{\Rightarrow} p_1 \overset{e}{\Rightarrow} q_1 \overset{\tau}{\Rightarrow} q}{p \overset{e}{\Rightarrow} q} \ (e \notin \{M\} \cup \mathcal{F})$$

As a running example, we consider a memory cell that stores a bit of infor-
mation and supports reading and writing operations, presented in a state-based
form in [11]. A state in this system maintains the current value of the memory
cell ($m = i$, for $i = 0, 1$), writing allows one to change this value, and reading
returns the stored value. Obviously, in this system the result of a reading depends
on the value stored in the cell. Thus, a property that one might associate with
this model is that the value read from the cell coincides with that of the last
writing performed in the system.

A potential fault in this scenario occurs when a cell unexpectedly loses its
charge, and its stored value turns into another one (e.g., it changes from 1 to
0 due to charge loss). A typical technique to deal with this situation is *redun-
dancy*: use three memory bits instead of one. Writing operations are performed
simultaneously on the three bits. Reading, on the other hand, returns the value
that is repeated at least twice in the memory bits; this is known as *voting*.

We take the following approach to model this system. Labels W_0, W_1, R_0, and
R_1 represent writing and reading operations. Specifically, W_0 (resp. W_1): writes
a zero (resp. one) in the memory. R_0 (resp. R_1): reads a zero (resp. one) from
the memory. Figure 1 depicts four transition systems. The leftmost one repre-
sents the nominal system for this example (denoted as A). The second one from
the left characterizes the nominal transition system augmented with masking
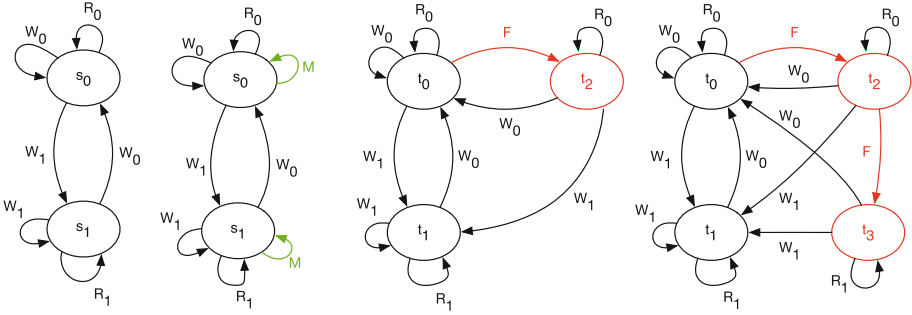
Fig. 1. Transition systems for the memory cell.

transitions, i.e., A^M. The third and fourth transition systems are fault-tolerant implementations of A, named A' and A'', respectively. Note that A' contains one fault, while A'' considers two faults. Both implementations use triple redundancy; intuitively, state t_0 contains the three bits with value zero and t_1 contains the three bits with value one. Moreover, state t_2 is reached when one of the bits was flipped (either 001, 010 or 100). In A'', state t_3 is reached after a second bit is flipped (either 011 or 101 or 110) starting from state t_0. It is straightforward to see that there exists a relation of masking fault-tolerance between A^M and A', as it is witnessed by the relation $\mathbf{M} = \{(s_0, t_0), (s_1, t_1), (s_0, t_2)\}$. It is a routine to check that $\mathbf{M}$ satisfies the conditions of Definition 1. On the other hand, there does not exist a masking relation between A^M and A'' because state t_3 needs to be related to state s_0 in any masking relation. This state can only be reached by executing faults, which are necessarily masked with M-transitions. However, note that, in state t_3, we can read a 1 (transition $t_3 \xrightarrow{R_1} t_3$) whereas, in state s_0, we can only read a 0.

3.3   Masking Simulation Game

We define a masking simulation game for two transition systems (the specification of the nominal system and its fault-tolerant implementation) that captures masking fault-tolerance. We first define the masking game graph where we have two players named by convenience the *refuter* (R) and the *verifier* (V).

Definition 3. *Let* $A = \langle S, \Sigma, E, s_0 \rangle$ *and* $A' = \langle S', \Sigma_{\mathcal{F}}, E'_W, s'_0 \rangle$. *The strong masking game graph* $\mathcal{G}_{A^M, A'} = \langle S^G, S_R, S_V, \Sigma^G, E^G, s_0{}^G \rangle$ *for two players is defined as follows:*

- $\Sigma^G = \Sigma_M \cup \Sigma_{\mathcal{F}}$
- $S^G = (S \times (\Sigma^1_M \cup \Sigma^2_{\mathcal{F}} \cup \{\#\}) \times S' \times \{R, V\}) \cup \{s_{err}\}$
- *The initial state is* $s_0^G = \langle s_0, \#, s'_0, R \rangle$, *where the refuter starts playing*
- *The refuter's states are* $S_R = \{(s, \#, s', R) \mid s \in S \wedge s' \in S'\} \cup \{s_{err}\}$
- *The verifier's states are* $S_V = \{(s, \sigma, s', V) \mid s \in S \wedge s' \in S' \wedge \sigma \in \Sigma^G \setminus \{M\}\}$

and E^G is the minimal set satisfying:

- $\{(s, \#, s', R) \xrightarrow{\sigma} (t, \sigma^1, s', V) \mid \exists\, \sigma \in \Sigma : s \xrightarrow{\sigma} t \in E\} \subseteq E^G$,
- $\{(s, \#, s', R) \xrightarrow{\sigma} (s, \sigma^2, t', V) \mid \exists\, \sigma \in \Sigma_{\mathcal{F}} : s' \xrightarrow{\sigma} t' \in E'\} \subseteq E^G$,
- $\{(s, \sigma^2, s', V) \xrightarrow{\sigma} (t, \#, s', R) \mid \exists\, \sigma \in \Sigma : s \xrightarrow{\sigma} t \in E\} \subseteq E^G$,
- $\{(s, \sigma^1, s', V) \xrightarrow{\sigma} (s, \#, t', R) \mid \exists\, \sigma \in \Sigma : s' \xrightarrow{\sigma} t' \in E'\} \subseteq E^G$,
- $\{(s, F^2, s', V) \xrightarrow{M} (t, \#, s', R) \mid \exists\, s \xrightarrow{M} t \in E^M\} \subseteq E^G$, *for any $F \in \mathcal{F}$*
- *If there is no outgoing transition from some state s then transitions $s \xrightarrow{\sigma} s_{err}$ and $s_{err} \xrightarrow{\sigma} s_{err}$ for every $\sigma \in \Sigma$, are added.*

The intuition of this game is as follows. The refuter chooses transitions of either the specification or the implementation to play, and the verifier tries to match her choice, this is similar to the bisimulation game [28]. However, when the refuter chooses a fault, the verifier must match it with a masking transition (M). The intuitive reading of this is that the fault-tolerant implementation masked the fault in such a way that the occurrence of this fault cannot be noticed from the users' side. R wins if the game reaches the error state, i.e., s_{err}. On the other hand, V wins when s_{err} is not reached during the game. (This is basically a reachability game [26]).

A *weak masking game graph* $\mathcal{G}^W_{A^M, A'}$ is defined in the same way as the strong masking game graph in Definition 3, with the exception that Σ_M and $\Sigma_{\mathcal{F}}$ may contain τ, and the set of labelled transitions (denoted as E^G_W) is now defined using the weak transition relations (i.e., E_W and E'_W) from the respective transition systems.

Figure 2 shows a part of the strong masking game graph for the running example considering the transition systems A^M and A''. We can clearly observe on the game graph that the verifier cannot mimic the transition $(s_0, \#, t_3, R) \xrightarrow{R^2_1} (s_0, R^2_1, t_3, V)$ selected by the refuter which reads a 1 at state t_3 on the fault-tolerant implementation. This is because the verifier can only read a 0 at state s_0. Then, the s_{err} is reached and the refuter wins.

As expected, there is a strong masking simulation between A and A' if and only if the verifier has a winning strategy in $\mathcal{G}_{A^M, A'}$.

Theorem 3. *Let $A = \langle S, \Sigma, E, s_0 \rangle$ and $A' = \langle S', \Sigma_{\mathcal{F}}, E', s'_0 \rangle$. $A \preceq_m A'$ iff the verifier has a winning strategy for the strong masking game graph $\mathcal{G}_{A^M, A'}$.*

By Theorems 2 and 3, the result replicates for weak masking game.

Theorem 4. *Let $A = \langle S, \Sigma \cup \{\tau\}, E, s_0 \rangle$ and $A' = \langle S', \Sigma_{\mathcal{F}} \cup \{\tau\}, E', s'_0 \rangle$. $A \preceq^w_m A'$ iff the verifier has a winning strategy for the weak masking game graph $\mathcal{G}^W_{A^M, A'}$.*

Using the standard properties of reachability games we get the following property.

Theorem 5. *For any A and A', the strong (resp. weak) masking game graph $\mathcal{G}_{A^M, A'}$ (resp. $\mathcal{G}^W_{A^M, A'}$) can be determined in time $O(|E^G|)$ (resp. $O(|E^G_W|)$).*
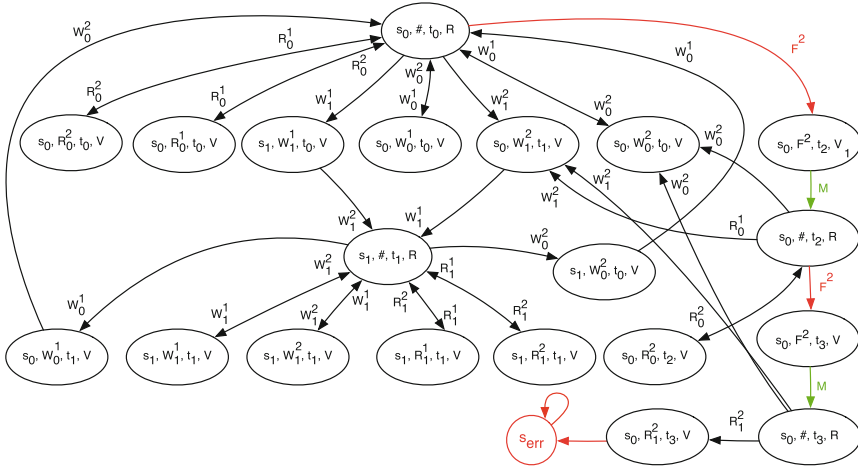
Fig. 2. Part of the masking game graph for memory cell model with two faults

The set of winning states for the refuter can be defined in a standard way from the error state [26]. We adapt ideas in [26] to our setting. For $i, j \geq 0$, sets U_i^j are defined as follows:

$$U_i^0 = U_0^j = \emptyset, \tag{1}$$
$$U_1^1 = \{s_{err}\},$$
$$U_{i+1}^{j+1} = \{v' \mid v' \in S_R \wedge post(v') \cap U_{i+1}^j \neq \emptyset\}$$
$$\cup \{v' \mid v' \in S_V \wedge post(v') \subseteq \bigcup_{j' \leq j} U_{i+1}^{j'} \wedge post(v') \cap U_{i+1}^j \neq \emptyset \wedge \pi_2(v') \notin \mathcal{F}\}$$
$$\cup \{v' \mid v' \in S_V \wedge post(v') \subseteq \bigcup_{i' \leq i, j' \leq j} U_{i'}^{j'} \wedge post(v') \cap U_i^j \neq \emptyset \wedge \pi_2(v') \in \mathcal{F}\}$$

then $U^k = \bigcup_{i \geq 0} U_i^k$ and $U = \bigcup_{k \geq 0} U^k$. Intuitively, the subindex i in U_i^k indicates that s_{err} is reach after at most $i - 1$ faults occurred. The following lemma is straightforwardly proven using standard techniques of reachability games [9].

Lemma 1. *The refuter has a winning strategy in $\mathcal{G}_{A^M, A'}$ (or $\mathcal{G}_{A^M, A'}^W$) iff $s_{init} \in U^k$, for some k.*

3.4   Quantitative Masking

In this section, we extend the strong masking simulation game introduced above with quantitative objectives to define the notion of masking fault-tolerance distance. Note that we use the attribute "quantitative" in a non-probabilistic sense.

Definition 4. *For transition systems A and A', the quantitative strong masking game graph $\mathcal{Q}_{A^M, A'} = \langle S^G, S_R, S_V, \Sigma^G, E^G, s_0^G, v^G \rangle$ is defined as follows:*

- $\mathcal{G}_{A^M,A'} = \langle S^G, S_R, S_V, \Sigma^G, E^G, s_0^G \rangle$ *is defined as in Definition 3,*
- $v^G(s \xrightarrow{e} s') = (\chi_{\mathcal{F}}(e), \chi_{s_{err}}(s'))$

where $\chi_{\mathcal{F}}$ is the characteristic function over set $\mathcal{F}$, returning 1 if $e \in \mathcal{F}$ and 0 otherwise, and $\chi_{s_{err}}$ is the characteristic function over the singleton set $\{s_{err}\}$.

Note that the cost function returns a pair of numbers instead of a single number. It is direct to codify this pair into a number, but we do not do it here for the sake of clarity. We remark that the *quantitative weak masking game graph* $\mathcal{Q}^W_{A^M,A'}$ is defined in the same way as the game graph defined above but using the weak masking game graph $\mathcal{G}^W_{A^M,A'}$ instead of $\mathcal{G}_{A^M,A'}$.

Given a quantitative strong masking game graph with the weight function v^G and a play $\rho = \rho_0 \sigma_0 \rho_1 \sigma_1 \rho_2 \ldots$, for all $i \geq 0$, let $v_i = v^G(\rho_i \xrightarrow{\sigma_i} \rho_{i+1})$. We define the *masking payoff function* as follows:

$$f_m(\rho) = \lim_{n \to \infty} \frac{\mathrm{pr}_1(v_n)}{1 + \sum_{i=0}^{n} \mathrm{pr}_0(v_i)},$$

which is proportional to the inverse of the number of masking movements made by the verifier. To see this, note that the numerator of $\frac{\mathrm{pr}_1(v_n)}{1 + \sum_{i=0}^{n} \mathrm{pr}_0(v_i)}$ will be 1 when we reach the error state, that is, in those paths not reaching the error state this formula returns 0. Furthermore, if the error state is reached, then the denominator will count the number of fault transitions taken until the error state. All of them, except the last one, were masked successfully. The last fault, instead, while attempted to be masked by the verifier, eventually leads to the error state. That is, the transitions with value $(1, \_)$ are those corresponding to faults. The others are mapped to $(0, \_)$. Notice also that if s_{err} is reached in v_n without the occurrence of any fault, the nominal part of the implementation does not match the nominal specification, in which case $\frac{\mathrm{pr}_1(v_n)}{1 + \sum_{i=0}^{n} \mathrm{pr}_0(v_i)} = 1$. Then, the refuter wants to maximize the value of any run, that is, she will try to execute faults leading to the state s_{err}. In contrast, the verifier wants to avoid s_{err} and then she will try to mask faults with actions that take her away from the error state. More precisely, the value of the quantitative strong masking game for the refuter is defined as $val_R(\mathcal{Q}_{A^M,A'}) = \sup_{\pi_R \in \Pi_R} \inf_{\pi_V \in \Pi_V} f_m(out(\pi_R, \pi_V))$. Analogously, the value of the game for the verifier is defined as $val_V(\mathcal{Q}_{A^M,A'}) = \inf_{\pi_V \in \Pi_V} \sup_{\pi_R \in \Pi_R} f_m(out(\pi_R, \pi_V))$. Then, we define the value of the quantitative strong masking game, denoted by $val(\mathcal{Q}_{A^M,A'})$, as the value of the game either for the refuter or the verifier, i.e., $val(\mathcal{Q}_{A^M,A'}) = val_R(\mathcal{Q}_{A^M,A'}) = val_V(\mathcal{Q}_{A^M,A'})$. This can be done because quantitative strong masking games are determined as we prove below in Theorem 6.

Definition 5. *Let A and A' be transition systems. The* strong masking distance *between A and A', denoted by $\delta_m(A, A')$ is defined as:* $\delta_m(A, A') = val(\mathcal{Q}_{A^M,A'})$.

We would like to remark that the *weak masking distance* δ_m^W is defined in the same way for the quantitative weak masking game graph $\mathcal{Q}^W_{A^M,A'}$. Roughly speaking, we are interesting on measuring the number of faults that can be

masked. The value of the game is essentially determined by the faulty and masking labels on the game graph and how the players can find a strategy that leads (or avoids) the state s_{err}, independently if there are or not silent actions.

In the following, we state some basic properties of this kind of games. As already anticipated, quantitative strong masking games are determined:

Theorem 6. *For any quantitative strong masking game $\mathcal{Q}_{A^M,A'}$ with payoff function f_m:*

$$\inf_{\pi_V \in \Pi_V} \sup_{\pi_R \in \Pi_R} f_m(out(\pi_R, \pi_V)) = \sup_{\pi_R \in \Pi_R} \inf_{\pi_V \in \Pi_V} f_m(out(\pi_R, \pi_V))$$

The value of the quantitative strong masking game can be calculated as stated below.

Theorem 7. *Let $\mathcal{Q}_{A^M,A'}$ be a quantitative strong masking game. Then, $\mathrm{val}(\mathcal{Q}_{A^M,A'}) = \frac{1}{w}$, with $w = \min\{i \mid \exists j : s_{init} \in U_i^j\}$, whenever $s_{init} \in U$, and $\mathrm{val}(\mathcal{Q}_{A^M,A'}) = 0$ otherwise, where sets U_i^j and U are defined in Eq. (1).*

Note that the sets U_i^j can be calculated using a bottom-up breadth-first search from the error state. Thus, the strategies for the refuter and the verifier can be defined using these sets, without taking into account the history of the play. That is, we have the following theorems:

Theorem 8. *Players R and V have memoryless winning strategies for $\mathcal{Q}_{A^M,A'}$.*

Theorems 6, 7, and 8 apply as well to $\mathcal{Q}_{A^M,A'}^W$. The following theorem states the complexity of determining the value of the two types of games.

Theorem 9. *The quantitative strong (weak) masking game can be determined in time $O(|S^G| + |E^G|)$ (resp. $O(|S^G| + |E_W^G|)$).*

Theorems 5 and 9 describe the complexity of solving the quantitative and standard masking games. However, in practice, one needs to bear in mind that $|S^G| = |S| * |S'|$ and $|E^G| = |E| + |E'|$, so constructing the game takes $O(|S|^2 * |S'|^2)$ steps in the worst case. Additionally, for the weak games, the transitive closure of the original model needs to be computed, which for the best known algorithm yields $O(\max(|S|, |S'|)^{2.3727})$ [30].

By using $\mathcal{Q}_{A^M,A'}^W$ instead of $\mathcal{Q}_{A^M,A'}$ in Definition 5, we can define the *weak masking distance* δ_m^W. The next theorem states that, if A and A' are at distance 0, there is a strong (or weak) masking simulation between them.

Theorem 10. *For any transition systems A and A', then (i) $\delta_m(A, A') = 0$ iff $A \preceq_m A'$, and (ii) $\delta_m^W(A, A') = 0$ iff $A \preceq_m^w A'$.*

This follows from Theorem 7. Noting that $A \preceq_m A$ (and $A \preceq_m^w A$) for any transition system A, we obtain that $\delta_m(A, A) = 0$ (resp. $\delta_m^W(A, A) = 0$) by Theorem 10, i.e., both distance are reflexive.

For our running example, the masking distance is 1/3 with a redundancy of 3 bits and considering two faults. This means that only one fault can be masked by this implementation. We can prove a version of the triangle inequality for our notion of distance.

Theorem 11. *Let* $A = \langle S, \Sigma, E, s_0 \rangle$, $A' = \langle S', \Sigma_{\mathcal{F}'}, E', s_0' \rangle$, *and* $A'' = \langle S'', \Sigma_{\mathcal{F}''}, E'', s_0'' \rangle$ *be transition systems such that* $\mathcal{F}' \subseteq \mathcal{F}''$. *Then* $\delta_m(A, A'') \leq \delta_m(A, A') + \delta_m(A', A'')$ *and* $\delta_m^W(A, A'') \leq \delta_m^W(A, A') + \delta_m^W(A', A'')$.

Reflexivity and the triangle inequality imply that both masking distances are directed semi-metrics [7,10]. Moreover, it is interesting to note that the triangle inequality property has practical applications. When developing critical software is quite common to develop a first version of the software taking into account some possible anticipated faults. Later, after testing and running of the system, more plausible faults could be observed. Consequently, the system is modified with additional fault-tolerant capabilities to be able to overcome them. Theorem 11 states that incrementally measuring the masking distance between these different versions of the software provides an upper bound to the actual distance between the nominal system and its last fault-tolerant version. That is, if the sum of the distances obtained between the different versions is a small number, then we can ensure that the final system will exhibit an acceptable masking tolerance to faults w.r.t. the nominal system.

4   Experimental Evaluation

The approach described in this paper has been implemented in a tool in Java called MaskD: Masking Distance Tool [1]. MaskD takes as input a nominal model and its fault-tolerant implementation, and produces as output the masking distance between them. The input models are specified using the guarded command language introduced in [3], a simple programming language common for describing fault-tolerant algorithms. More precisely, a program is a collection of processes, where each process is composed of a collection of actions of the style: $Guard \rightarrow Command$, where $Guard$ is a boolean condition over the actual state of the program and $Command$ is a collection of basic assignments. These syntactical constructions are called actions. The language also allows user to label an action as internal (i.e., τ actions). Moreover, usually some actions are used to represent faults. The tool has several additional features, for instance it can print the traces to the error state or start a simulation from the initial state.

We report on Table 1 the results of the masking distance for multiple instances of several case studies. These are: a Redundant Cell Memory (our running example), N-Modular Redundancy (a standard example of fault-tolerant system [27]), a variation of the Dining Philosophers problem [13], the Byzantine Generals problem introduced by Lamport et al. [22], and the Bounded Retransmission Protocol (a well-known example of fault-tolerant protocol [16]).

Some words are useful to interpret the results. For the case of a 3 bit memory the masking distance is 0.333, the main reason for this is that the faulty model in the worst case is only able to mask 2 faults (in this example, a fault is an unexpected change of a bit value) before failing to replicate the nominal behaviour (i.e. reading the majority value), thus the result comes from the definition of masking distance and taking into account the occurrence of two

faults. The situation is similar for the other instances of this problem with more redundancy.

N-Modular-Redundancy consists of N systems, in which these perform a process and that results are processed by a majority-voting system to produce a single output. Assuming a single perfect voter, we have evaluated this case study for different numbers of modules. Note that the distance measures for this case study are similar to the memory example.

For the dining philosophers problem we have adopted the odd/even philosophers implementation (it prevents from deadlock), i.e., there are $n-1$ *even* philosophers that pick the right fork first, and 1 *odd* philosopher that picks the left fork first. The fault we consider in this case occurs when an even philosopher behaves as an odd one, this could be the case of a byzantine fault. For two philosophers the masking distance is 0.5 since a single fault leads to a deadlock, when more philosophers are added this distance becomes smaller.

Another interesting example of a fault-tolerant system is the Byzantine generals problem, introduced originally by Lamport et al. [22]. This is a consensus problem, where we have a general with $n-1$ lieutenants. The communication between the general and his lieutenants is performed through messengers. The general may decide to attack an enemy city or to retreat; then, he sends the order to his lieutenants. Some of the lieutenants might be traitors. We assume that the messages are delivered correctly and all the lieutenants can communicate directly with each other. In this scenario they can recognize who is sending a message. Faults can convert loyal lieutenants into traitors (byzantines faults). As a consequence, traitors might deliver false messages or perhaps they avoid sending a message that they received. The loyal lieutenants must agree on attacking or retreating after $m+1$ rounds of communication, where m is the maximum numbers of traitors.

The Bounded Retransmission Protocol (BRP) is a well-known industrial case study in software verification. While all the other case studies were treated as toy

Table 1. Results of the masking distance for the case studies.

Case Study	Redundancy	Masking Distance	Time
Memory	3 bits	0.333	0.7s
	5 bits	0.25	1.5s
	7 bits	0.2	27s
	9 bits	0.167	34m33s
N-Modular Redundancy	3 modules	0.333	0.3s
	5 modules	0.25	0.5s
	7 modules	0.2	31.7s
	9 modules	0.167	115m
Philosophers	2 phils	0.5	0.3s
	3 phils	0.333	0.6s
	4 phils	0.25	7.1s
	5 phils	0.2	13m.53s
Byzantines	3 generals	0.5	0.5s
	4 generals	0.333	2s

Case Study	Redundancy	Masking Distance	Time
BRP(1)	1 retransm.	0.333	1.2s
	3 retransm.	0.2	1.4s
	5 retransm.	0.143	1.5s
	7 retransm.	0.111	2.1s
BRP(3)	1 retransm.	0.333	5.5s
	3 retransm.	0.2	14.9s
	5 retransm.	0.143	1m28s
	7 retransm.	0.111	4m40s
BRP(5)	1 retransm.	0.333	6.7s
	3 retransm.	0.2	32s
	5 retransm.	0.143	1m51s
	7 retransm.	0.111	6m35s

examples and analyzed with δ_m, the BRP was modeled closer to the implementation following [16], considering the different components (sender, receiver, and models of the channels). To analyze such a complex model we have used instead the weak masking distance δ_m^W. We have calculated the masking distance for the bounded retransmission protocol with 1, 3 and 5 chunks, denoted BRP(1), BRP(3) and BRP(5), respectively. We observe that the distance values are not affected by the number of chunks to be sent by the protocol. This is expected because the masking distance depends on the redundancy added to mask the faults, which in this case, depends on the number of retransmissions.

We have run our experiments on a MacBook Air with Processor 1.3 GHz Intel Core i5 and a memory of 4 Gb. The tool and case studies for reproducing the results are available in the tool repository.

5   Related Work

In recent years, there has been a growing interest in the quantitative generalizations of the boolean notion of correctness and the corresponding quantitative verification questions [4,6,17,18]. The framework described in [6] is the closest related work to our approach. The authors generalize the traditional notion of simulation relation to three different versions of simulation distance: *correctness*, *coverage*, and *robustness*. These are defined using quantitative games with *discounted-sum* and *mean-payoff* objectives, two well-known cost functions. Similarly to that work, we also consider distances between purely discrete (non-probabilistic, untimed) systems. Correctness and coverage distances are concerned with the nominal part of the systems, and so faults play no role on them. On the other hand, robustness distance measures how many unexpected errors can be performed by the implementation in such a way that the resulting behavior is tolerated by the specification. So, it can be used to analyze the resilience of the implementation. Note that, robustness distance can only be applied to correct implementations, that is, implementations that preserve the behavior of the specification but perhaps do not cover all its behavior. As noted in [6], bisimilarity sometimes implies a distance of 1. In this sense a greater grade of robustness (as defined in [6]) is achieved by pruning critical points from the specification. Furthermore, the errors considered in that work are transitions mimicking the original ones but with different labels. In contrast to this, in our approach we consider that faults are injected into the fault-tolerant implementation, where their behaviors are not restricted by the nominal system. This follows the idea of model extension in fault-tolerance where faulty behavior is added to the nominal system. Further, note that when no faults are present, the masking distance between the specification and the implementation is 0 when they are bisimilar, and it is 1 otherwise. It is useful to note that robustness distance of [6] is not reflexive. We believe that all these definitions of distance between systems capture different notions useful for software development, and they can be used together, in a complementary way, to obtain an in-depth evaluation of fault-tolerant implementations.

6   Conclusions and Future Work

In this paper, we presented a notion of masking fault-tolerance distance between systems built on a characterization of masking tolerance via simulation relations and a corresponding game representation with quantitative objectives. Our framework is well-suited to support engineers for the analysis and design of fault-tolerant systems. More precisely, we have defined a computable masking distance function such that an engineer can measure the masking tolerance of a given fault-tolerant implementation, i.e., the number of faults that can be masked. Thereby, the engineer can measure and compare the masking fault-tolerance distance of alternative fault-tolerant implementations, and select one that fits best to her preferences.

There are many directions for future work. We have only defined a notion of fault-tolerance distance for masking fault-tolerance, similar notions of distance can be defined for other levels of fault-tolerance like failsafe and non-masking. Also, we have focused on non-quantitative models. However, metrics defined on probabilistic models, where the rate of fault occurrences is explicitly represented, could give a more accurate notion of fault tolerance.

References

1. MaskD: Masking distance tool. https://github.com/lputruele/MaskD
2. Apt, K.R., Grädel, E.: Lectures in Game Theory for Computer Scientists, 1st edn. Cambridge University Press, New York (2011)
3. Arora, A., Gouda, M.: Closure and convergence: a foundation of fault-tolerant computing. IEEE Trans. Softw. Eng. **19**(11), 1015–1027 (1993)
4. Boker, U., Chatterjee, K., Henzinger, T.A., Kupferman, O.: Temporal specifications with accumulative values. ACM Trans. Comput. Log. **15**(4), 27:1–27:25 (2014)
5. Castro, P.F., D'Argenio, P.R., Demasi, R., Putruele, L.: Measuring masking fault-tolerance. CoRR, abs/1811.05548 (2018)
6. Cerný, P., Henzinger, T.A., Radhakrishna, A.: Simulation distances. Theor. Comput. Sci. **413**(1), 21–35 (2012)
7. Charikar, M., Makarychev, K., Makarychev, Y.: Directed metrics and directed graph partitioning problems. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, 22–26 January 2006, pp. 51–60 (2006)
8. de Alfaro, L., Faella, M., Stoelinga, M.: Linear and branching system metrics. IEEE Trans. Softw. Eng. **35**(2), 258–273 (2009)
9. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theor. Comput. Sci. **386**(3), 188–217 (2007)
10. de Alfaro, L., Majumdar, R., Raman, V., Stoelinga, M.: Game refinement relations and metrics. Log. Methods Comput. Sci. **4**(3:7) (2008)
11. Demasi, R., Castro, P.F., Maibaum, T.S.E., Aguirre, N.: Simulation relations for fault-tolerance. Formal Aspects Comput. **29**(6), 1013–1050 (2017)
12. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labelled Markov processes. Theor. Comput. Sci. **318**(3), 323–354 (2004)

13. Dijkstra, E.W.: Hierarchical ordering of sequential processes. Acta Informatica **1**(2), 115–138 (1971)
14. Fernandez, J.-C., Mounier, L.: "On the fly" verification of behavioural equivalences and preorders. In: Larsen, K.G., Skou, A. (eds.) CAV 1991. LNCS, vol. 575, pp. 181–191. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55179-4_18
15. Gärtner, F.C.: Fundamentals of fault-tolerant distributed computing in asynchronous environments. ACM Comput. Surv. **31**, 1–26 (1999)
16. Groote, J.F., van de Pol, J.: A bounded retransmission protocol for large data packets. In: Wirsing, M., Nivat, M. (eds.) AMAST 1996. LNCS, vol. 1101, pp. 536–550. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0014338
17. Henzinger, T.A.: From Boolean to quantitative notions of correctness. In: Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, 17–23 January 2010, pp. 157–158 (2010)
18. Henzinger, T.A.: Quantitative reactive modeling and verification. Comput. Sci. - R&D **28**(4), 331–344 (2013)
19. Henzinger, T.A., Majumdar, R., Prabhu, V.S.: Quantifying similarities between timed systems. In: Pettersson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 226–241. Springer, Heidelberg (2005). https://doi.org/10.1007/11603009_18
20. Hsueh, M.-C., Tsai, T.K., Iyer, R.K.: Fault injection techniques and tools. Computer **30**(4), 75–82 (1997)
21. Iyer, R.K., Nakka, N., Gu, W., Kalbarczyk, Z.: Fault injection. In: Encyclopedia of Software Engineering, pp. 287–299 (2010)
22. Lamport, L., Shostak, R.E., Pease, M.C.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
23. Larsen, K.G., Fahrenberg, U., Thrane, C.R.: Metrics for weighted transition systems: axiomatization and complexity. Theor. Comput. Sci. **412**(28), 3358–3369 (2011)
24. Martin, D.A.: The determinacy of blackwell games. J. Symb. Log. **63**(4), 1565–1581 (1998)
25. Milner, R.: Communication and Concurrency. Prentice-Hall Inc., Upper Saddle River (1989)
26. Jurdziński, M.: Algorithms for solving parity games. In: Apt, K.R., Grädel, E. (eds.) Lectures in Game Theory for Computer Scientist, chap. 3, pp. 74–95. Cambridge University Press, New York (2011)
27. Shooman, M.L.: Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design. Wiley, Hoboken (2002)
28. Stirling, C.: Bisimulation, modal logic and model checking games. Log. J. IGPL **7**(1), 103–124 (1999)
29. Thrane, C.R., Fahrenberg, U., Larsen, K.G.: Quantitative analysis of weighted transition systems. J. Log. Algebraic Program. **79**(7), 689–703 (2010)
30. Williams, V.V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, 19–22 May 2012 (2012)