Retaining the Probabilities in Probabilistic Testing Theory

Sonja Georgievska and Suzana Andova

Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
s.georgievska@tue.nl, s.andova@tue.nl

Abstract. This paper considers the probabilistic may/must testing theory for processes having external, internal, and probabilistic choices. We observe that the underlying testing equivalence is too strong and distinguishes between processes that are observationally equivalent. The problem arises from the observation that the classical compose-and-schedule approach yields unrealistic overestimation of the probabilities, a phenomenon that has been recently well studied from the point of view of compositionality (de Alfaro/Henzinger/Jhala 2001, Cheung/Lynch/Segala/Vaandrager 2006), in the context of randomized protocols (Chatzikokolakis/Palamidessi 2007), and in probabilistic model checking (Giro/D'Argenio/Ferrer Fioriti 2009). To that end, we propose a new testing theory, aiming at preserving the probability information in a parallel context. The resulting testing equivalence is insensitive to the exact moment the internal and the probabilistic choices occur. We also give an alternative characterization of the testing preorder as a probabilistic ready-trace preorder.

1 Introduction

The testing theory for concurrent processes [7, 13] is based on the criterion that two processes are equivalent iff they cannot be distinguished when interacting with their environment, which is an arbitrary process itself. The aim is to obtain the coarsest congruence for parallel composition by the definition of the relation. It turned out that, for a broad class of processes, the testing equivalence [7, 13] and the failures equivalence [1] coincide [20]. Both theories, [1] and [7, 13] gave prominence to the notion of "unobservability" of the exact moment in which a process makes an internal choice.

Originally, only qualitative properties were in the focus of theories for reactive processes. In many applications with unreliable components, however, it is more useful to prove that a system will deadlock less than 0.1% of the time, rather than never. This is one of the reasons why probabilistic choice, as a refinement of internal choice, was introduced in process semantics, in addition to action choice and standard internal choice. To be able to reason about the probabilistic behavior of systems, *schedulers* were adopted, which resolve the nondeterminism and yield fully probabilistic systems.

L. Ong (Ed.): FOSSACS 2010, LNCS 6014, pp. 79–93, 2010.

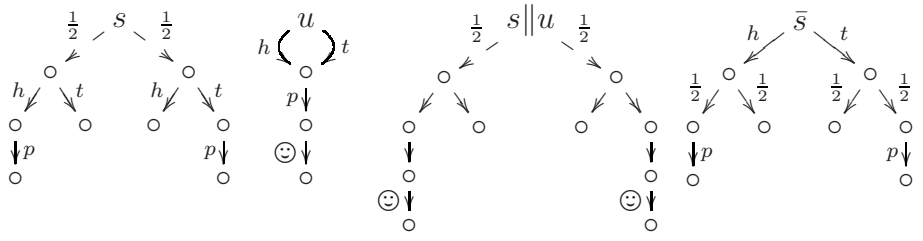

Fig. 1. The coin-flipping machine and the guessing user

Originally, schedulers were monolithic: processes were composed and schedulers were applied to the composed system. With this reasoning, an extension of the may/must testing theory for the case with probabilistic choice was defined [24]. However, it was soon observed that the compose-and-schedule approach could yield unrealistic overestimation of the probabilistic behaviour of the composed system [17, 19, 22]. We explain this phenomenon via the examples that follow.

The gambling machine. Consider a system consisting of a machine and a user, that communicate via a menu of two buttons "head" and "tail" positioned at the machine. The machine first makes a fair choice (i.e. flips a coin) whether to give a prize if "head" is chosen or if "tail" is chosen. Then the user is allowed to choose "head" or "tail" by pressing the appropriate button. If the user chooses the right outcome, a prize is awarded. Note that by no means the user could have detected the machine's choice beforehand. The machine can be modeled by process graph s in Fig. 1. Thus, in half of the machine runs, it offers a prize after the "head" button has been pressed (out of the two-button menu "head" and "tail"), while in the other half of the runs it offers a prize after the "tail" button has been pressed (out of the two-button menu "head" and "tail"). The user can be modeled by process graph u in Fig. 1. Sometimes she would press "head" and sometimes "tail"; however, she is "happy" (denoted by action ☺) iff, after pressing a button, a prize follows. Note that the user makes her choice based on the options the machine offers; in this example the menu consists of two options.

Let the user and the machine interact, by synchronizing on their common actions, except on action ☺. In terms of testing theory [7], process s is tested with test u. By means of probability theory, it can be calculated that the probability that the user has guessed the output of flipping is $\frac{1}{2}$. That is, the probability of a ☺ action being reported is $\frac{1}{2}$. However, most of the existing approaches for probabilistic testing, in particular probabilistic may/must testing [8, 9, 15, 21, 23, 24], do not give this answer. As we mentioned above, in order to compute the probability of ☺ being reported, the probabilistic may/must testing uses schedulers. These schedulers have access to the nodes of the graph of the synchronized process.

Consider the synchronization $s \parallel u$ represented by the process graph in Fig. 1, where actions are hidden after they have synchronized (τ-labels are omitted in the figure). Each scheduler resolves the choice in the nondeterministic nodes of $s \parallel u$ and yields a fully probabilistic process graph. For $s \parallel u$ in Fig. 1 there are four possible schedulers, which yield the following set of probabilities with which s passes the test u: $\{0, \frac{1}{2}, 1\}$. We can see that, because the power of the schedulers is unrestricted, unrealistic upper and lower bounds for the probabilities are obtained. Observe that this happens due to the effect of "cloning" the synchronization nondeterminism. The choice between synchronizing on h or t has been "cloned" in both futures after the probabilistic choice in $s \parallel u$. In this way, when resolving the nondeterminism in $s \parallel u$, a scheduler assumes that the user has unrealistic power to *see* the result of the coin-flipping *before* guessing.

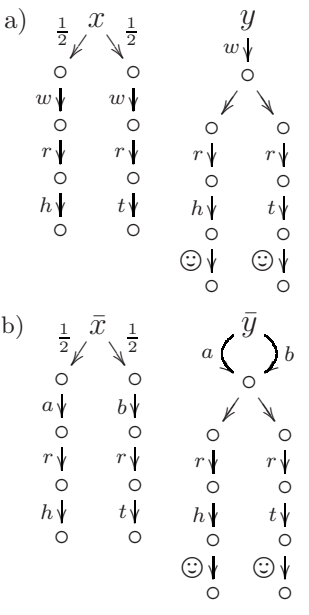

Fig. 2. The coin tosser and result-guesser game: a) fair play; b) unfair play

Consider now process graph $\bar{s}$ in Fig. 1. To the user $\bar{s}$ may as well represent the behaviour of the coin-flipping machine – the user cannot see whether the machine flips the coin *before* or *after* making the "head or tail" offers. According to the user, the machine acts as specified as long as she is able to guess the result in half of the cases. In fact, both schedulers defined by the probabilistic may/must testing, when applied to $\bar{s} \parallel u$ yield exactly probability $\frac{1}{2}$ of reporting a ☺ action. Consequently, none of the approaches in [8, 9, 15, 21, 23, 24] equate processes s and $\bar{s}$: when tested with u, they produce different bounds for the probabilities of reporting ☺[1].

The coin tosser – result guesser game. Consider the following game. Player x tosses a fair coin and hides the outcome. Player y guesses the outcome of the tossing and writes it down. While he is writing the result down, player x waits (i.e. he may write down something meaningless). Then, they both agree to reveal their outcomes, i.e. x to uncover the coin and y to show what he has written. Players x and y are modeled in Fig. 2a. Obviously, the probability that the second player guesses correctly, i.e. reports ☺, equals $\frac{1}{2}$. However, the schedulers applied to the resulting graph for the synchronization of both players give the set of probabilities $\{0, \frac{1}{2}, 1\}$, thus, suggesting that there is a strategy such that player y can always guess the result correctly. On the other hand, if process $x' = w.r.(h \oplus_{\frac{1}{2}} t)$ is synchronized with y, both schedulers applied to the resulting process graph suggest that the probability of reporting a ☺ action is exactly $\frac{1}{2}$. Hence, in probabilistic may/must testing theory processes x and x' cannot

[1] If we ignore the probabilities, processes s and $\bar{s}$ are testing-equivalent by [7].

be equated, and therefore prefix does not distribute over internal probabilistic choice. In other words, the internal (probabilistic) choice is "observable". [2]

It is important to note that, if the players are as in Fig. 2b, i.e. the coin tosser reveals the outcome by offering different actions for synchronization (a or b), then the guesser can surely guess the result: he would make his guess depending on the action on which both players previously synchronized (a or b).

1.1 Contributions

The testing preorder relations defined in [23, 24] were shown to be branching time (simulations) in [9, 15, 18]. Thus, distribution of prefix over internal/probabilistic choice is not allowed. This implies that if we are about to combine external, classical internal and internal probabilistic choice, for verification purposes we can only rely on equivalences that inspect the internal structure of the process graphs. Nevertheless, the most discouraging fact is that in a parallel composition the probability information might be lost, as the previous examples show. This questions the reasons for adding probabilities in the model initially.

The main contribution of this paper is testing semantics in the style of [7] for processes exhibiting external, internal and probabilistic choices that does not yield over-estimated or under-estimated probabilities with which a process passes a test. To achieve this, we propose a novel method for labeling the internal transitions that arise in the parallel composition of the process and the test. The usage of the labels solves the problem with the "cloned nondeterminism" and allows us to compute the probabilities with which a process passes a test. Moreover, unlike the schedulers-based approaches [23, 24], our approach makes the resolution of the nondeterminism state-independent, which, we believe, is more appropriate for testing-based semantics.

The technical contributions of the paper are the following. In our model the internal transitions with the same node of origin have different labels which serve as identifiers for the transitions in a parallel context (Sec. 3). Having this, we first define resolution of the internal nondeterminism in a process (Sec. 4.1). Then, when testing a process with a test, given a resolution of the internal nondeterminism of the process, we define how the synchronization nondeterminism and the internal nondeterminism of the test are resolved (Sec. 4.2). Our solution avoids overestimation and underestimation of the probabilities with which the process passes the test. Based on the testing semantics, we define a preorder relation between processes from which it follows that one process implements another iff the former contains "less" nondeterminism (Sec. 4.2). Finally, we characterize the testing preorder as a probabilistic ready-trace preorder relation (Sec. 5). The latter reveals that, indeed, our testing equivalence is insensitive to the exact moment of occurrence of internal/probabilistic choices.

Due to lack of space and to avoid technical complications while presenting the ideas and the results, in this paper we restrict to finitely-branching, divergence-free, though recursive processes and to finite tests.

[2] Variants of this problem were initially pointed in [17, 19, 22] and treated in [3, 5, 6, 11].

2 Preliminaries

In this paper we use the Bayesian definition of probability because it is more suitable than the measure-theoretic one for our definition of ready-trace preorder.

Bayesian probability For a set A, 2^A denotes its power-set. The following definitions are taken from [16].

We consider a sample space, Ω, consisting of points called *elementary events*. Selection of a particular $a \in \Omega$ is referred to as an "a has occurred". An *event* $A \subseteq \Omega$ is a set of elementary events. $A, B, C, \ldots$ range over events. An event A *has occurred* iff a has occurred for some $a \in A$. Let $A_1, A_2, \ldots$ be a sequence of events and C be an event. The members of the sequence are *exclusive given* C, if whenever C has occurred no two of them can occur together, that is, if $A_i \cap A_j \cap C = \emptyset$ whenever $i \neq j$. C is called a *conditioning* event. If the conditioning event is Ω, then "given Ω" is omitted.

For certain pairs of events A and B, a real number $P(A|B)$ is defined and called the *probability* of A given B. These numbers satisfy the following axioms:

A1: $0 \leq P(A|B) \leq 1$ and $P(A|A) = 1$.
A2: If the events in $\{A_i\}_{i=1}^{\infty}$ are exclusive given B, then $P(\cup_{i=1}^{\infty} A_i \mid B) = \sum_{i=1}^{\infty} P(A_i|B)$.
A3: $P(C|A \cap B) \cdot P(A|B) = P(A \cap C|B)$.

For $P(A|\Omega)$ we simply write $P(A)$.

3 Model

We define our model of process graphs and operations on them. Presuppose a finite set of actions $\mathcal{A}$ and a countable set of labels $\mathcal{L}$ such that $\mathcal{A} \cap \mathcal{L} = \emptyset$. A process graph has three types of transitions (edges), representing action, internal, and probabilistic transitions; a state (node) can perform only one type of transitions. The action transitions are decorated with actions from $\mathcal{A}$, while the internal transitions with labels from $\mathcal{L}$. No two action transitions with the same state of origin are labeled the same and no two internal transitions with the same state of origin are labeled the same. Moreover, if two states have transitions each with the same label, then the sets of all labels of their outgoing (internal) transitions coincide. All states are reachable from a designated initial state via zero or more transitions.

Definition 1 (Process graph). *A* process graph *(or simply process) is a tuple* $\mathcal{P} = (S_{\mathcal{A}}, S_p, S_\tau, \rightarrow, \twoheadrightarrow, \dashrightarrow, r)$, *where*

- $S_{\mathcal{A}}, S_p, S_\tau$ *are finite sets of* action, probabilistic, and nondeterministic states *respectively,*
- $r \in S_{\mathcal{A}} \cup S_p \cup S_\tau$ *is the* initial *state,*
- $\rightarrow \subseteq S_{\mathcal{A}} \times \mathcal{A} \times (S_{\mathcal{A}} \cup S_p \cup S_\tau)$ *is an* action transition relation *such that* $(s, a, t) \in \rightarrow$ *and* $(s, a, t') \in \rightarrow$ *implies* $t = t'$,

- $\twoheadrightarrow \subseteq S_\tau \times \mathcal{L} \times (S_\mathcal{A} \cup S_p \cup S_\tau)$ *is an* internal transition relation *such that*
  - $(s,l,t) \in \twoheadrightarrow$ *and* $(s,l,t') \in \twoheadrightarrow$ *implies* $t = t'$, *and*
  - *if* $(s,l,t) \in \twoheadrightarrow$ *and* $(s',l,t') \in \twoheadrightarrow$ *for some* $l \in \mathcal{L}$ *and states* t,t', *then, for every* $k \in \mathcal{L}$, $(s,k,q) \in \twoheadrightarrow$ *for some* q *iff* $(s',k,q') \in \twoheadrightarrow$ *for some* q',
- $\dashrightarrow \subseteq S_p \times (0,1] \times (S_\mathcal{A} \cup S_p \cup S_\tau)$ *is a* probabilistic transition relation *such that* $(s,\pi,t) \in \dashrightarrow$ *and* $(s,\rho,t) \in \dashrightarrow$ *implies* $\pi = \rho$, *and for all* $s \in S$, $\sum_{(s,\pi,t)\in\dashrightarrow} \pi = 1$, *and*
- *for every state* $s \in S_\mathcal{A} \cup S_p \cup S_\tau$, *there exists a sequence* $\{(s_i,\alpha_i,s_{i+1})\}_{i=1}^n$ *such that* $s_1 = r, s_{n+1} = s$, *and for every* $i \in \{1,\dots n\}$, (s_i,α_i,s_{i+1}) *belongs to one of the transition relations.*

The set of all process graphs is denoted by $\mathcal{G}$. A process graph is usually named by its initial state. We write $s \xrightarrow{a} t$ rather than $(s,a,t) \in \rightarrow$, $s \overset{l}{\twoheadrightarrow} t$ rather than $(s,l,t) \in \twoheadrightarrow$, and $s \overset{\pi}{\dashrightarrow} t$ rather than $(s,\pi,t) \in \dashrightarrow$ (or $s \dashrightarrow t$ if the value of π is irrelevant in the context). We write $s \xrightarrow{a}$ to denote that there exists an action transition $s \xrightarrow{a} s'$ for some $s' \in S$, where by S we denote $S_\mathcal{A} \cup S_p \cup S_\tau$. Given a process s and an action $a \in \mathcal{A}$, by s_a we denote the process (if exists) for which $s \xrightarrow{a} s_a$.

For a finite index set I, by $s \xrightarrow{a_i} \{s_i\}_{i\in I}$ (resp. $s \overset{\pi_i}{\dashrightarrow} \{s_i\}_{i\in I}$, $s \overset{\tau_i}{\twoheadrightarrow} \{s_i\}_{i\in I}$) we denote that there exist transitions $\{s \xrightarrow{a_i} s_i\}_{i\in I}$ (resp. $\{s \overset{\pi_i}{\dashrightarrow} s_i\}_{i\in I}$, $\{s \overset{\tau_i}{\twoheadrightarrow} s_i\}_{i\in I}$) and s has no other outgoing transitions.

We define the parametric operators $\oplus$, $\sqcap$, $\Box$ [3] on process graphs.

Definition 2. *Let* I *be a finite index set and* $\{s_i\}_{i\in I}$ *be a set of process graphs. For* $\{\pi_i\}_{i\in I} \subset (0,1]$ *such that* $\sum_{i\in I}\pi_i = 1$, $\{\tau_i\}_{i\in I} \subset \mathcal{L}$ *and* $\{a_i\}_{i\in I} \subseteq \mathcal{A}$, *the parametric operators* $\oplus$, $\sqcap$, $\Box$ *on process graphs are defined as follows:* $\oplus_{i\in I}\pi_i s_i$ *(resp.* $\Box_{i\in I}\ a_i s_i$, $\sqcap_{i\in I}\tau_i s_i$ *) is constructed by creating a new state* s, *a set of disjoint copies* $\{s'_i\}_{i\in I}$ *of the process graphs* $\{s_i\}_{i\in I}$, *and transitions* $\{s \overset{\pi_i}{\dashrightarrow} s'_i\}_{i\in I}$ *(resp.* $\{s \xrightarrow{a_i} s'_i\}_{i\in I}$, $\{s \overset{\tau_i}{\twoheadrightarrow} s'_i\}_{i\in I}$ *). The constant* δ *denotes a process that has only one state,* $\delta \in S_\mathcal{A}$, *and no transitions.*

Given a process graph $\mathcal{P} = (S_\mathcal{A}, S_p, S_\tau, \rightarrow, \twoheadrightarrow, \dashrightarrow, r)$, let $I\colon S_\mathcal{A} \mapsto 2^\mathcal{A}$ be a function such that, for all $a \in \mathcal{A}, s \in S_\mathcal{A}$, it holds $a \in I(s)$ iff $s \xrightarrow{a}$. $I(s)$ is called the *menu* of s. Intuitively, for $s \in S_\mathcal{A}$, $I(s)$ is the set of actions that process s can perform initially.

A *finite* process graph is a process graph in which only finite paths exist. A *divergence-free* process graph is a process graph without infinite sequences of probabilistic or internal transitions. To simplify the presentation, throughout the paper we assume that processes are divergence-free.

4 Testing Preorder

In this section we define a testing preorder relation in the style of [7] for the model introduced in Sec. 3. We first define the notion of unfolding a process and

[3] These operators are not to be confused with the CSP operators; however, we choose them so that they have the same intuitive meaning.

the notion of resolving internal nondeterminism in a process. They are needed in Sec. 4.2, where we define the testing preorder.

4.1 Resolving Internal Nondeterminism

The internal nondeterminism in a process is resolved by appropriately assigning probability distributions to the internal choices, i.e. assigning probabilities to the labels of the internal transitions. We first define a function that unfolds a process to a certain depth and relabels the internal transitions. Then we define how the probabilities are assigned.

Let $\mathbb{E}$ denote the set of equations of type $\sum_{i\in I} x_i = 1$ such that I is a finite index set and $\{x_i\}_{i\in I} \subset \mathcal{L}$. For a given set of non-negative numbers I, by $\max I$ we denote the maximum of the numbers in I, if $I \neq \emptyset$, or 0, if $I = \emptyset$. We slightly abuse the notation $\oplus$ and by $\oplus_{i\in I}\pi_i(s_i, \mathcal{C}_i)$ we denote the pair $(\oplus_{i\in I}\pi_i s_i, \cup_{i\in I}\mathcal{C}_i)$, for a given finite index set I, a set of process graphs $\{s_i\}_{i\in I}$, and a set $\{\mathcal{C}_i\}_{i\in I}$ such that $\mathcal{C}_i \subset \mathbb{E}$ for $i \in I$. Similarly for $\Box$ and $\sqcap$.

We next define the unfolding recursive function. The unfolding "depth" equals the maximal length of a sequence of observable actions that the unfolded process could perform.

Definition 3. *The partial function* $\mathrm{U}\colon \mathcal{G} \times 2^{\mathbb{E}} \times \mathbb{N} \mapsto \mathcal{G} \times 2^{\mathbb{E}}$*, called* unfolding*, is defined as* [4]

$$\mathrm{U}(s, \mathcal{C}, m) =$$

$$\begin{cases} \oplus_{i\in I}\, \pi_i \mathrm{U}(s_i, \mathcal{C}, m), & \text{if } m > 0, s \xrightarrow{\pi_i} \{s_i\}_{i\in I} \\ \sqcap_{i\in I} \tau_i^{n+1} \mathrm{U}(s_i, \mathcal{C} \cup \{\sum_{i\in I} \tau_i^{n+1} = 1\}, m), & \text{if } m > 0, s \overset{\tau_i}{\twoheadrightarrow} \{s_i\}_{i\in I}, \\ & n = \max\{k | \sum_{i\in I} \tau_i^k = 1 \in \mathcal{C}\} \\ \Box_{i\in I}\, a_i \mathrm{U}(s_i, \mathcal{C}, m-1), & \text{if } m > 0, s \xrightarrow{a_i} \{s_i\}_{i\in I} \\ (\delta, \mathcal{C}) & \text{otherwise} \end{cases}$$

where the labels $\{\tau_i^{n+1}\}$ *are fresh labels.*

By unfolding of a process graph a finite tree is obtained. The labels of the internal transitions in the obtained tree are fresh with respect to the labels in the original process graph. Every time a state with outgoing internal transitions labeled with the set $\{\tau_i\}_{i\in I}$ is to be unfolded, a fresh label set $\{\tau_i^{n+1}\}_{i\in I}$ is used for labeling this particular internal choice in the tree. This label set, in fact, contains implicit information , n, about the number of times a state in the original graph with the same label set $\{\tau_i\}_{i\in I}$ has been unfolded up to that moment. The set of equations obtained by unfolding are used later for assigning probabilities to the labels.

Intuitively, if in the original process graph one internal choice happens in the future of another, although they have the same labels, they are different from

[4] In the definition of the function, the index n in τ_i^n is not to be confused with n^{th} power of τ_i; we would denote the latter with $(\tau_i)^n$.

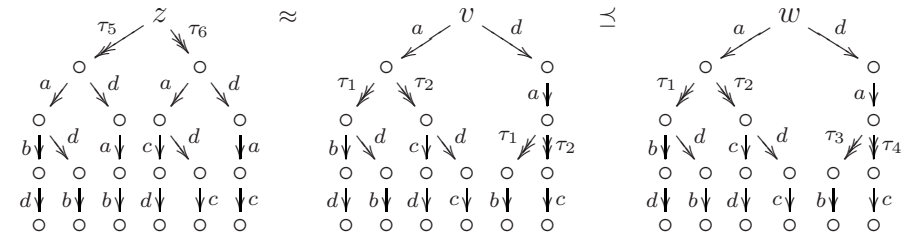

Fig. 3. Relations between several processes; processes z and v are obtained by interleaving $\tau_5(ab) \sqcap \tau_6(ac)$, resp. $a(\tau_1 b \sqcap \tau_2 c)$, with action d

each other. Therefore, they are given different labels in the unfolded tree. On the contrary, if choices with the same set of labels are placed "in parallel", as those in process v in Fig. 3, then they represent the same choice. Namely, since v in Fig. 3 is the parallel composition of process $q = a(\tau_1 b \sqcap \tau_2 c)$ and action d, and process q does not synchronize on action d, the internal choice of q cannot be influenced by action d. Consequently, both internal choices that appear in the graph of process v must be resolved in the same manner.

Given a tree that results from unfolding a process, we define how probabilities are assigned to the labels of the internal transitions.

Definition 4. *Let s be a process graph, $m \geq 0$, $\mathrm{U}(s, \emptyset, m) = (\bar{s}, \mathcal{C})$, and $\mathcal{L}_{/\mathcal{C}}$ be the set of all variables that appear in $\mathcal{C}$. Let $\lambda\colon \mathcal{L}_{/\mathcal{C}} \mapsto [0,1]$ be a function assigning values to the variables in $\mathcal{L}_{/\mathcal{C}}$ respecting the constraints $\mathcal{C}$. We call λ a* resolution *of $\mathcal{C}$. Given λ, let s^m be the process graph obtained when every transition $s_i \overset{\tau_i^k}{\twoheadrightarrow} s_i'$ that belongs to the process graph $\bar{s}$ is replaced by $s_i \overset{\lambda(\tau_i^k)}{\dashrightarrow} s_i'$, if $\lambda(\tau_i^k) \neq 0$, or erased if $\lambda(\tau_i^k) = 0$. We call s^m a m-*resolution *of process s.*

Intuitively, s^m is the process that results when the internal choices in $\bar{s}$ have been replaced by probabilistic choices with labels given by λ.

4.2 Definition of the Testing Preorder

A *test* T, as usual, is a finite process graph given as a tree, such that, for a symbol $\omega \notin \mathcal{A}$, there may exist transitions $s \xrightarrow{\omega}$ for some states s of T, denoting success. Additionally, we assume that no two labels of the internal transitions of the test are the same (we justify this assumption below). By $\mathcal{T}$ we denote the set of all tests. Given a test T, by $\mathsf{length}(T)$ we denote the maximal length of a sequence of observable actions which T can perform before performing ω.

In the previous subsection we defined how the internal nondeterminism of a process is resolved. However, two other types of nondeterminism arise when a process is being tested: first, the nondeterminism with respect to the action on which the process and the test synchronize, if there are more than one candidate-actions for synchronization at the moment (*synchronization* nondeterminism), and, second, the internal nondeterminism of the test.

Regarding the synchronization nondeterminism, note that the resolution must not depend on the internal transitions of the processes; otherwise the problem with overestimated probabilities, as discussed in the example on the gambling machine in Sec. 1, would remain. Second, we must not *underestimate* the power of the entity that resolves the synchronization nondeterminism, which can be even more hazardous [5]. For example, consider again the machine s, the user u, and their synchronization $s\|u$ in Fig. 1, described in Sec. 1. [6] Obviously, in order to keep the probability of $\frac{1}{2}$ with which s passes test u, we need to "remember" that both internal choices in $s\|u$ are resolved in the same way. Therefore, if the process and the test do not have a history of synchronization, the way the synchronization nondeterminism is resolved should depend on the set of candidate-actions for synchronization and on nothing else. On the other hand, as the process and the test proceed interacting, the entity that resolves the synchronization nondeterminism may actually "remember" its history of resolution. Therefore, we also take into account this history in the resolution of the current synchronization nondeterminism – otherwise, we would risk underestimation of the probabilities with which the process passes the test.

Concerning the test, we assume that all labels of internal transitions that it contains are distinct. That is, we restrict to the subset of tests that are most powerful and have "full control" over their nondeterminism. In fact, if two processes are not distinguished by this subset of tests, then they should not be distinguished by the rest of the (less powerful) tests either. Now, to avoid underestimation of the probabilities, we must not exclude the possibility that the test itself is the entity that resolves the synchronization nondeterminism (as in the gambling machine). Therefore, the test can also take into account the history of resolving the synchronization nondeterminism, when making decisions on resolving its internal nondeterminism. For example, consider the test $a(\tau_1 d\omega \sqcap \tau_2 c\omega) \;\square\; bc\omega$ and the process $\frac{1}{2}(ad \;\square\; b) \oplus \frac{1}{2}(ac)$. When testing the process, the test can choose transition τ_1 if action a was performed out of the options a and b, and the transition τ_2 if a was the only option on the menu. In this way, the chances to report success increase.

We now formalize the above discussions. By $\mathbb{P}$ we denote the set of all polynomials with variable names in the labels set $\mathcal{L}$. For a given finite index set I, a set of polynomials $\{\phi_i\}_{i\in I} \subset \mathbb{P}$, a set $\{\mathcal{C}_i\}_{i\in I}$ with $\mathcal{C}_i \subset \mathbb{E}$ for $i \in I$, and a set $\{\alpha_i\}_{i\in I}$ of scalars or variables in $\mathcal{L}$, by $\sum_{i\in I} \alpha_i(\phi_i, \mathcal{C}_i)$ we denote the pair $(\sum_{i\in I} \alpha_i\phi_i, \cup_{i\in I}\mathcal{C}_i)$.

Let $\mathcal{G}_{\not\to} \subset \mathcal{G}$ be the set of all process graphs that contain no internal transitions. We next define recursively the function for computing the result of testing a process in $\mathcal{G}_{\not\to} \subset \mathcal{G}$ with a test.

[5] With overestimated probabilities we might fail to verify a correct protocol [3, 12], but with underestimated probabilities we might claim a protocol to be correct, even if it is not.

[6] Here actually the user itself resolves the synchronization nondeterminism, but in general this is not the case.

Definition 5. *The partial function* $\mathsf{R}\colon \mathcal{G}_{\not\to} \times \mathcal{T} \times \mathcal{L} \times 2^{\mathbb{E}} \mapsto \mathbb{P} \times 2^{\mathbb{E}}$ *is defined as*

$$\mathsf{R}(s,T,l,\mathcal{C}) =$$

$$\begin{cases} (1,\mathcal{C}), & \text{if } T \xrightarrow{\omega} \\ \sum_{i\in I} \pi_i \cdot \mathsf{R}(s_i,T,l,\mathcal{C}), & \text{if } s \xrightarrow{\pi_i} \{s_i\}_{i\in I}, T \not\xrightarrow{\omega} \\ \sum_{i\in I} \pi_i \cdot \mathsf{R}(s,T_i,l,\mathcal{C}), & \text{if } T \xrightarrow{\pi_i} \{T_i\}_{i\in I}, s \not\to \\ \sum_{i\in I} \tau_i^l \cdot \mathsf{R}(s,T_i,l,\mathcal{C} \cup \{\sum_{i\in I} \tau_i^l = 1\}), & \text{if } T \xrightarrow{\tau_i} \{T_i\}_{i\in I}, s \not\to \\ \sum_{a\in K} \tau_{(a,K)}^l \mathsf{R}(s_a,T_a,\tau_{(a,K)}^l,\mathcal{C} \cup \{\sum_{a\in K} \tau_{(a,K)}^l = 1\}), & \text{for } K = I(s) \cap I(T) \neq \emptyset \\ (0,\mathcal{C}) & \text{otherwise.} \end{cases}$$

where the labels $\{\tau_i^l\}$ *and* $\{\tau_{(a,K)}^l\}$ *are fresh labels.*

Definition 6. *Let* ε *be a special label that cannot appear in any process or test,* $s \in \mathcal{G}_{\not\to}$ *and* T *be a test. Let* $\mathsf{R}(s,T,\varepsilon,\emptyset) = (\phi,\mathcal{C})$ *and* $\mathcal{L}_{/\phi}$ *be the set of all variables that appear in* ϕ*. Let* $\lambda\colon \mathcal{L}_{/\phi} \mapsto [0,1]$ *be a function assigning values to the variables in* $\mathcal{L}_{/\phi}$ *respecting the constraints* $\mathcal{C}$*. We call* λ *a* resolution *of* $s|T$*. The value of the polynomial* ϕ *for the values of its variables given by* λ *is denoted by* $\Pr(s,T,\lambda)$*.*

Intuitively, for a process s without internal transitions, $\Pr(s,T,\lambda)$ is the probability with which s passes test T, given that the synchronization nondeterminism and the internal nondeterminism of the test are resolved by λ.

We now define the testing preorder relation. Intuitively, a process s *implements* a process t iff for every test T it holds that, for every resolution of the internal nondeterminism in s, there exists a resolution of the internal nondeterminism in t, such that the options for resolving the synchronization nondeterminism and the internal nondeterminism in the test are the same for both processes when tested with T, and, moreover, for every possible resolution of the synchronization nondeterminism and the internal nondeterminism in the test, the probability with which s passes T is equal to the probability with which t passes T.

Definition 7. *Let* s *and* t *be two processes.* s implements t*, denoted by* $s \preceq_{\mathcal{T}} t$*, iff for every test* T *with* $\mathsf{length}(T) = m$ *and for every* m*-resolution* s^m *of* s*, there exists an* m*-resolution* t^m *of* t*, such that* $\mathsf{R}(s^m,T,\varepsilon,\emptyset) = \mathsf{R}(t^m,T,\varepsilon,\emptyset)$*, and, given an arbitrary resolution* λ *of* $s^m|T$*, it holds that* $\Pr(s^m,T,\lambda) = \Pr(t^m,T,\lambda)$*.*

In general, process s implements process t iff s contains "less" internal nondeterminism than process t.

Definition 8. *Processes* s *and* t *are* testing-equivalent*, denoted by* $s \approx_{\mathcal{T}} t$*, iff* $s \preceq_{\mathcal{T}} t$ *and* $t \preceq_{\mathcal{T}} s$*.*

Examples. Consider processes s and $\bar{s}$ and test u in Fig. 1. Neither of them has internal nondeterminism. When test u is applied to s and $\bar{s}$, the set of options for resolving the synchronization nondeterminism is the same for both processes. Moreover, for every resolution of the synchronization nondeterminism, the probability with which s and $\bar{s}$ pass test u is $\frac{1}{2}$. Consider now process x and test y in

Fig. 2. Process x has no internal nondeterminism and there is no synchronization nondeterminism. No matter how y resolves its internal nondeterminism, the probability with which x passes y is $\frac{1}{2}$.

Remark 1. Note that Definitions 3 and 5 can be merged into one definition for the process graph resulting from testing a process with a test. The process would have only probabilistic, internal and ω-transitions and the labels of the internal transitions would be created as the resulting tree is being created, according to Def. 3 and Def. 5. We separated the definitions for clarity and because Def. 3 is also needed in Sec. 5 and it simplifies the proof of Theorem 1.

5 Characterizing the Testing Preorder as a Probabilistic Ready-Trace Preorder

In this section we characterize the testing preorder with a probabilistic ready-trace preorder.

Definition 9 (Ready trace). *A* ready trace of length n *is a sequence* $\mathcal{O} = (M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$ *where* $M_i \in 2^{\mathcal{A}}$ *for all* $i \in \{1, 2, \ldots, n\}$ *and* $a_i \in M_i$ *for all* $i \in \{1, 2, \ldots, n-1\}$.

We assume that the observer has the ability to observe the actions that the process performs, together with the menus out of which actions are chosen. Intuitively, a ready trace $\mathcal{O} = (M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$ can be observed if the initial menu is M_1, then action $a_1 \in M_1$ is performed, then the next menu is M_2, then action $a_2 \in M_2$ is performed and so on, until the observing ends at a point when the menu is M_n.

Next, given a process $s \in \mathcal{G}_{\not\rightarrow}$, i.e. without internal transitions, we define a process $s_{(M,a)}$. Intuitively, $s_{(M,a)}$ is the process that s becomes, assuming that menu M was offered to s and action a was performed. For example, for process s in Fig. 1, $s_{(\{h,t\},h)} = \frac{1}{2}p \oplus \frac{1}{2}\delta$.

Suppose $s \stackrel{\pi_1}{\dashrightarrow} s_1 \stackrel{\pi_2}{\dashrightarrow} s_2 \ldots \stackrel{\pi_n}{\dashrightarrow} s_n$ and $s_n \in S_{\mathcal{A}}$. Denote the product $\pi_1\pi_2\cdots\pi_n$ by π. We write $s \stackrel{\pi}{\Rightarrow} s_n$ rather than $s \stackrel{\pi_1}{\dashrightarrow} s_1 \stackrel{\pi_2}{\dashrightarrow} s_2 \ldots \stackrel{\pi_n}{\dashrightarrow} s_n$.

Definition 10. *Let* $s \in \mathcal{G}_{\not\rightarrow}$. *Let* $M \subseteq \mathcal{A}$, $a \in M$ *be such that* $I(s) = M$ *if* $s \in S_{\mathcal{A}}$, *or otherwise there exists a transition* $s \Rightarrow s'$ *such that* $I(s') = M$. *The process graph* $s_{(M,a)}$ *is obtained from* s *in the following way:*

- *if* $s \stackrel{a}{\rightarrow} s_a$ *then* $s_{(M,a)} = s_a$;
- *if* $s \dashrightarrow$ *then* $s_{(M,a)} \equiv \oplus_{i \in I} \frac{\pi_i}{\pi} s'_i$, *where* $\{s'_i\}_{i \in I}$ *are all process graphs s.t. for* $i \in I$ *there exists a sequence of transitions* $s \stackrel{\pi_i}{\Rightarrow} s_i \stackrel{a}{\rightarrow} s'_i$ *s.t.* $I(s_i) = M$, *and* $\pi = \sum_{s \stackrel{\pi_i}{\Rightarrow} s_i, I(s_i)=M} \pi_i$.

Definition 11. *Let* $(M_1, a_1, M_2, a_2, \ldots, M_{n-1}, a_{n-1}, M_n)$ *be a ready trace of length* n *and* $s \in \mathcal{G}_{\not\rightarrow}$. *Functions* $P^1_s(M)$ *and* $P^n_s(M_n | M_1, a_1, \ldots M_{n-1}, a_{n-1})$ *(for* $n > 1$*) are defined in the following way:*

$$P_s^1(M) = \begin{cases} \sum_{s \xrightarrow{\pi} s'} \pi \cdot P_{s'}^1(M) & \text{if } s \in S_p, \\ 1 & \text{if } s \in S_{\mathcal{A}},\ I(s) = M, \\ 0 & \text{otherwise.} \end{cases}$$

$$P_s^2(M_2|M_1, a_1) = \begin{cases} P^1_{s_{(M_1,a_1)}}(M_2) & \text{if } P_s^1(M_1) > 0, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$P_s^n(M_n|M_1, a_1, \ldots, a_{n-1}) = \begin{cases} P^{n-1}_{s_{(M_1,a_1)}}(M_n|M_2, a_2, \ldots, a_{n-1}) & \text{if } P_s^1(M_1) > 0, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Let the sample space consist of all possible menus and let $s \in \mathcal{G}_{\not\to}$. Function $P_s^1(M)$ can be interpreted as the probability that menu M is observed when process s starts executing. Let the sample space consist of all ready traces of length n. Function $P_s^n(M_n|M_1, a_1, \ldots M_{n-1}, a_{n-1})$ can be interpreted as the probability of the event $\{(M_1, a_1, \ldots, M_{n-1}, a_{n-1}, M_n)\}$, given the event $\{(M_1, a_1, \ldots M_{n-1}, a_{n-1}, X) : X \in 2^{\mathcal{A}}\}$, when observing ready traces of process s. These probabilities are well defined, i.e. they satisfy axioms A1-A3 of Sec. 2.

Definition 12. *Let s and t be two processes. We say s* implements *t* w.r.t. ready traces *(notation $s \preceq_{\mathcal{O}} t$) iff for every $n \geq 0$ and every n-resolution $\bar{s}$ of s, there exists a n-resolution $\bar{t}$ of t such that for all $k \leq n$ and for all ready traces $(M_1, a_1, \ldots M_k)$,*

- *$P^1_{\bar{s}}(M_1) = P^1_{\bar{t}}(M_1)$ and*
- *if $k > 1$, then $P^k_{\bar{s}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ is defined if and only if $P^k_{\bar{t}}(M_k|M_1, a_1, \ldots M_{k-1}, a_{k-1})$ is defined, and, in case they are both defined, they are equal.*

Informally, a process s implements a process t iff, for every n-resolution $\bar{s}$ of the nondeterminism in s, there is an n-resolution $\bar{t}$ of the nondeterminism in t such that for every ready trace $(M_1, a_1, M_2, a_2, \ldots M_k)$ of length $k \leq n$, the probability to observe M_k, under the condition that the sequence $(M_1, a_1, M_2, a_2, \ldots M_{k-1}, a_{k-1})$ was previously observed, is defined at the same time for both $\bar{s}$ and $\bar{t}$, and, moreover, in case both probabilities are defined, they coincide.

Definition 13. *Let s and t be two processes. s and t are ready-trace-equivalent, denoted by $s \approx_{\mathcal{O}} t$, iff $s \preceq_{\mathcal{O}} t$ and $t \preceq_{\mathcal{O}} s$.*

Examples. Processes s and $\bar{s}$ in Fig.1 are ready-trace equivalent. Processes z and v in Fig. 3 are ready-trace equivalent and they both implement process w in the same figure. Processes z and v can be actually seen as an interleaving of processes $\tau_5 ab \sqcap \tau_6 ac$, resp. $a(\tau_1 b \sqcap \tau_2 c)$, none of which can recognize action d, with action d, while process w has "full control" over its nondeterminism. Fig. 4 also represents relations between several processes.

Theorem 1. *Let s and t be two processes. $s \preceq_{\mathcal{O}} t$ if and only if $s \preceq_{\mathcal{T}} t$.*

Proof. See [10].

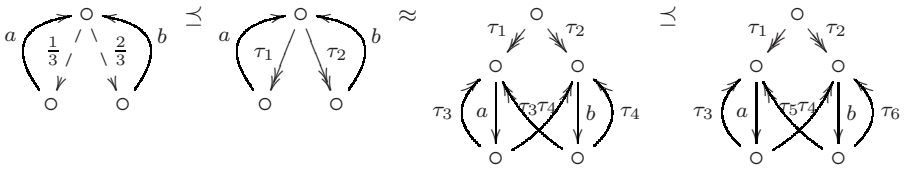

Fig. 4. Relations between several recursive processes

6 Related Work

Testing preorders in the style of [7] for systems exhibiting external, internal and internal probabilistic choices were first introduced in [24], where centralized (monolithic) schedulers, as discussed in Sec. 1, were used. These preorders were later extensively studied and characterized as simulations [8, 9, 15, 21, 23]. Motivated to allow distribution of prefix over internal (probabilistic) choice, as in [7, 13, 14], proposals for testing semantics were given in [19] and [2]. In these approaches all probabilistic choices in a process are resolved at the beginning of the execution. Thus, internal choices are "pushed" downwards and replicated, having as a side-effect loss of the probability information and loss of idempotence of internal choice.

The phenomenon of overestimated probabilities that results from system analysis using centralized schedulers was first observed in [17] and also pointed at in [19, 22]; however, it started being treated relatively recently.

The first paper to introduce distributed schedulers for concurrent systems to overcome the problem with the super-powerful centralized schedulers, discussed in Sec. 1, is [6], for a setting and parallel composition that is rather different than ours. The motivation is to allow compositionality for the trace distributions inclusion (see e.g. [22]). Reference [5] also treats the compositionality problem for trace distributions inclusion for reactive-generative probabilistic systems, by exploiting distributed schedulers.

The first paper to introduce the finer reasoning of testing semantics [7] under restricted schedulers is [3]. It introduces an *explicit* scheduler, which communicates with the process via labels: two nodes in the process are indistinguishable to a scheduler if they have the same labels. For example, the probabilities with which process s (Fig. 1) passes test u, and whether processes s and $\bar{s}$ can be equated, depend on the labeling system.

Unlike the previous approaches for solving the problem with overestimated probabilities in concurrent processes, the "schedulers" that we define do not use information about the states of the processes. We believe that this is essential if the equivalence relation is based on the notion of observability or testing. On the other hand, taking into consideration all possible resolutions of internal nondeterminism seems to be unavoidable when defining preorder relations between probabilistic-nondeterministic systems. For example, even in statistical *black-box* testing (see [4]), one needs to consider all possible schedulers in order to come up with a reasonable notion of trace distribution inclusion.

7 Conclusion

We defined a testing preorder relation in the style of [7] for processes exhibiting internal, external, and probabilistic internal nondeterminism. The goal of the testing semantics was to preserve the probability information in a parallel context and to produce realistic estimates of the probabilities with which a process passes a given test, based on the information that both entities exchange. To our knowledge, this is the first testing semantics of its kind that tackles this problem. We also showed that the testing preorder relation can be characterized with a probabilistic ready-trace preorder, which is easier to grasp from an observer's point of view. From this characterization it easily follows that the equivalence relation is insensitive to the exact moment of occurrence of internal and probabilistic choices.

In the future we plan to extend our work to divergent and infinitely-branching processes, to consider recursive tests and to study the preorder relation defined here in a process algebraic setting. It would be also interesting to apply the testing preorder for verification purposes.

Acknoledgements. We are grateful to the anonymous reviewers for FoSSaCS'10, whose detailed comments improved the final version of this paper.

References

1. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. Journal of ACM 31(3), 560–599 (1984)
2. Cazorla, D., Cuartero, F., Valero, V., Pelayo, F.L., Pardo, J.J.: Algebraic theory of probabilistic and nondeterministic processes. Journal of Logic and Algebraic Programming 55(1-2), 57–103 (2003)
3. Chatzikokolakis, K., Palamidessi, C.: Making random choices invisible to the scheduler. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 42–58. Springer, Heidelberg (2007)
4. Cheug, L., Stoelinga, M.I.A., Vaandrager, F.W.: A testing scenario for probabilistic processes. Journal of ACM 54(6), 29:1–29:45 (2007)
5. Cheung, L., Lynch, N., Segala, R., Vaandrager, F.: Switched PIOA: Parallel composition via distributed scheduling. Theoret. Comp. Science 365(1-2), 83–108 (2006)
6. de Alfaro, L., Henzinger, T., Jhala, R.: Compositional methods for probabilistic systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001)
7. De Nicola, R., Hennessy, M.C.B.: Testing equivalences for processes. Theoret. Comp. Science 34, 83–133 (1984)
8. Deng, Y., van Glabbeek, R., Hennessy, M., Morgan, C.: Testing finitary probabilistic processes (extended abstract). In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009 - Concurrency Theory. LNCS, vol. 5710, pp. 274–288. Springer, Heidelberg (2009)
9. Deng, Y., van Glabbeek, R.J., Hennessy, M., Morgan, C.: Characterising testing preorders for finite probabilistic processes. Logical Methods in Computer Science 4(4:4), 1–33 (2008)

10. Georgievska, S., Andova, S.: Retaining the probabilities in probabilistic testing theory. Technical Report (to appear, 2010), http://www.win.tue.nl/~sgeorgie/general_testing.pdf
11. Giro, S., D'Argenio, P.: On the expressive power of schedulers in distributed probabilistic systems. In: Proc. QAPL 2009. ENTCS (2009) (to appear)
12. Giro, S., D'Argenio, P., Ferrer Fioriti, L.M.: Partial order reduction for probabilistic systems: A revision for distributed schedulers. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009 - Concurrency Theory. LNCS, vol. 5710, pp. 338–353. Springer, Heidelberg (2009)
13. Hennessy, M.: Algebraic Theory of Processes. MIT Press, Cambridge (1988)
14. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall, Englewood Cliffs (1985)
15. Jonsson, B., Wang, Y.: Testing preorders for probabilistic processes can be characterized by simulations. Theoret. Comp. Science 282(1), 33–51 (2002)
16. Lindley, D.V.: Introduction to Probability and Statistics from a Bayesian Viewpoint. Cambridge University Press, Cambridge (1980)
17. Lowe, G.: Representing nondeterministic and probabilistic behaviour in reactive processes. Technical Report PRG-TR-11-93, Oxford Univ. Comp. Labs (1993)
18. Lynch, N., Segala, R., Vaandrager, F.: Observing branching structure through probabilistic contexts. SIAM Journal on Computing 37(4), 977–1013 (2007)
19. Morgan, C., McIver, A., Seidel, K., Sanders, J.W.: Refinement-oriented probability for CSP. Formal Aspects of Computing 8(6), 617–647 (1996)
20. De Nicola, R.: Extensional equivalences for transition systems. Acta Informatica 24(2), 211–237 (1987)
21. Palmeri, M.C., De Nicola, R., Massink, M.: Basic observables for probabilistic testing. In: Proc. QEST 2007, pp. 189–200. IEEE Computer Society, Los Alamitos (2007)
22. Segala, R.: Modeling and verification of randomized distributed real-time systems. PhD thesis. MIT (1995)
23. Segala, R.: Testing probabilistic automata. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 299–314. Springer, Heidelberg (1996)
24. Wang, Y., Larsen, K.G.: Testing probabilistic and nondeterministic processes. In: Proceedings of the IFIP TC6/WG6.1 Twelth International Symposium on Protocol Specification, Testing and Verification XII, pp. 47–61 (1992)