Scalar Outcomes Suffice for Finitary Probabilistic Testing

Yuxin Deng[1⋆], Rob van Glabbeek[2,1], Carroll Morgan[1⋆] & Chenyi Zhang[1,2]

[1] School of Comp. Sci. and Eng., University of New South Wales, Sydney, Australia
[2] National ICT Australia, Locked Bag 6016, Sydney, NSW 1466, Australia

Abstract. The question of equivalence has long vexed research in concurrency, leading to many different denotational- and bisimulation-based approaches; a breakthrough occurred with the insight that tests expressed within the concurrent framework itself, based on a special "success action," yield equivalences that make only inarguable distinctions.
When probability was added, however, it seemed necessary to extend the testing framework beyond a direct probabilistic generalisation in order to remain useful. An attractive possibility was the extension to *multiple* success actions that yielded *vectors* of real-valued outcomes.
Here we prove that such vectors are unnecessary when processes are *finitary*, that is finitely branching and finite-state: single *scalar* outcomes are just as powerful. Thus for finitary processes we can retain the original, simpler testing approach and its direct connections to other naturally scalar-valued phenomena.

1 Introduction

The theory of testing of De Nicola & Hennessy [4] yields equivalences making only inarguable distinctions: two processes are may-testing inequivalent iff there is a context, built with parallel composition and hiding or restriction operators, in which one of them *might* do a visible action but the other definitely can not; they are must-testing inequivalent iff there is a context in which one *must* do a visible action, but the other might never do any. This reduces a complex phenomenon to a scalar- (in fact Boolean-) valued outcome.

Wang & Larsen [21] generalised this theory in a straightforward way to processes with probabilistic and nondeterministic choice, again yielding only distinctions that are hard to argue with: two processes are may-testing inequivalent iff there is a context in which, in the best case (when resolving nondeterministic choice), one of them might do a visible action with some probability p whereas the other falls short of that. They are must-testing inequivalent iff there is a context in which, in the worst case, one must do a visible action with probability p, whereas the other might fall short of that.

Wang & Larsen ended with the question of finding denotational characterisations of may- and must testing; for non-trivial examples this is necessary to show the equivalence of two processes. The question is still open today, although Jonsson & Wang [9] have found a denotational characterisation in the special

⋆ We acknowledge the support of the Australian Research Council Grant DP034557.

case of may-testing for processes without internal actions.

Meanwhile, progress has been made elsewhere for notions of testing that are seemingly more powerful than Wang & Larsen's. Most notably, Segala [17] found denotational characterisations of variants of may- and must testing that employ multiple success actions instead of a single one, and consequently yield vectors of real-valued outcomes instead of scalars. Earlier, Jonsson, Ho-Stuart & Wang [8] characterised variants that employ so-called "reward testing," and that consider non-probabilistic test processes only.

It follows immediately from the definitions that Segala's vector-based testing is *at least* as powerful as the scalar testing of Wang & Larsen, while reward testing sits in between. Unfortunately, the possibility that these extended notions of testing are strictly *more* powerful suggests that the argument above, that testing equivalences make no unwarranted distinctions, might not apply to them.[3]

In this paper we show that in fact the argument *does* apply, at least for finitary processes, where we prove all three notions to be equally powerful. This is a fundamental prerequisite for a stable notion of program equivalence, a core concept for rigorous software engineering.

2 Probabilistic Testing of Probabilistic Automata

2.1 Probabilistic Structures and Notational Conventions

We write $f.s$ instead of $f(s)$ for function application, with left association so that $f.g.x$ means $(f(g))(x)$.

- A *discrete probability distribution* over a set X is a function $\mu \in X \to [0,1]$ with $\mu.X = 1$, where for subset X' of X we define $\mu.X' := \sum_{x \in X'} \mu.x$. We write $\overline{X}$ for the set of all such distributions over X.
- The *point-* or *Dirac distribution* $\overline{x}$ assigns probability one to $x \in X$.
- The *support* $\lceil \mu \rceil$ of distribution μ is the set of elements x such that $\mu.x \neq 0$.
- Given $p \in [0,1]$ and distributions $\mu, \zeta \in \overline{X}$, the *weighted average* $\mu \,{}_p\!\oplus\, \zeta \in \overline{X}$ is the distribution defined $(\mu \,{}_p\!\oplus\, \zeta).x := p \times \mu.x + (1-p) \times \zeta.x$.
- Given some function $f \in X \to \mathbb{R}$ (a *random variable*), its *expected value* $\mu.f$ over distribution μ is the weighted average $\sum_{x \in X} (\mu.x \times f.x)$.[4]
- Given a function $f \in X \to Y$, we write $f.\mu$ for the *image distribution* in $\overline{Y}$ formed by applying f to a distribution $\mu \in \overline{X}$: for element $y \in Y$ we define $f.\mu.y := \mu.\{x \in X \mid f.x = y\}$.
- The *product* of two discrete probability distributions μ, μ' over X, X' is the distribution $\mu \times \mu'$ over $X \times X'$ defined $(\mu \times \mu').(x, x') := \mu.x \times \mu'.x'$.

2.2 Probabilistic Automata and their Resolutions

In this paper we employ probabilistic automata [18] as representatives of a class of models that treat both probabilistic and nondeterministic choice [20, 5, 21, 6].

[3] However, Stoelinga & Vaandrager [19] offer evidence that the distinctions of Segala's vector-based may-testing are observable by repeating experiments many times.

[4] This extends to any linear vector space, in particular to functions in $X \to \mathbb{R}^N$.

Definition 1. A *probabilistic automaton* is a tuple $\mathrm{M} = (M, m^\circ, E, I, \mathcal{T})$ where

- M is a set of *states*,
- $m^\circ \in \overline{M}$ is a distribution of *start states*,
- E and I are disjoint sets of *external-* and *internal actions* respectively and
- $\mathcal{T} \in M \to \mathcal{P}(\Sigma \times \overline{M})$ is the *transition relation*, where $\Sigma := E \cup I$.

Automaton M is *fully probabilistic* if from each state $m \in M$ there is at most one outgoing transition, *i.e.* if the set $\mathcal{T}.m$ contains at most one element. If $\mathcal{T}.m$ is finite for all $m \in M$, then M is said to be *finitely branching*. An *execution sequence* of M is an alternating sequence of states and actions $m_0, \alpha_1, m_1, \alpha_2, \cdots$, either infinite or ending in state m_n, such that $m_0 \in \lceil m^\circ \rceil$ and for all $i > 0$ (and $i \leq n$ if finite) we have $\exists (\alpha_i, \mu_i) \in \mathcal{T}.m_{i-1}$ with $m_i \in \lceil \mu_i \rceil$. The execution sequence is *maximal* if either it is infinite or $\mathcal{T}.m_n = \emptyset$. ¶

From here on we use "automaton" to mean "probabilistic automaton."

Any automaton can be "resolved" into fully probabilistic automata as follows.

Definition 2. A *resolution* of an automaton $\mathrm{M} = (M, m^\circ, E, I, \mathcal{T})$ is a fully probabilistic automaton $\mathrm{R} = (R, r^\circ, E, I, \mathcal{T}')$ such that there is a *resolving function* $f \in R \to M$ with

- $f.r^\circ = m^\circ$, *equivalent initialisations*
- if $\mathcal{T}'.r = \{(\alpha, \mu)\}$ then $(\alpha, f.\mu) \in \mathcal{T}.(f.r)$ and *compatible choice taken*
- if $\mathcal{T}'.r = \emptyset$ then $\mathcal{T}.(f.r) = \emptyset$ *liveness preserved*

for any $r \in R$. ¶ [5]

A resolution has as its main effect the choosing in any state of a single outgoing transition from all available ones; but f can be non-injective, so that the choice can vary between different departures from that state, depending *e.g.* on the history of states and actions that led there. Further, since a single state of M can be "split" into a distribution over several states of R, all mapped to it by f, probabilistic *interpolation* between distinct choices is obtained automatically.[6]

Fig. 1 illustrates the history-dependent choice with states $r_{1,4}$ both mapped to m_1; it illustrates interpolation with states $r_{2,4}$ both mapped to m_1.

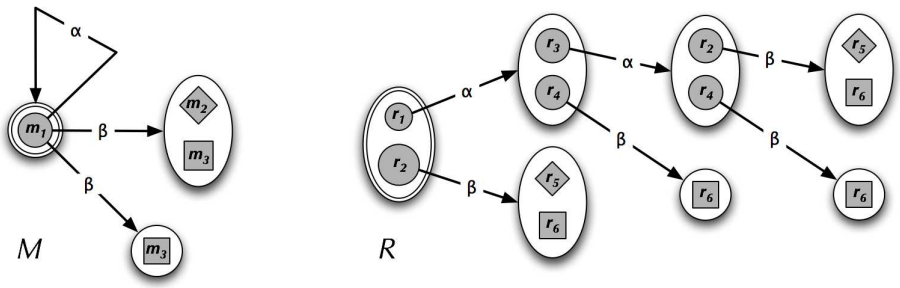
2.3 Probabilities of Action Occurrences in Resolutions

For a fully probabilistic automaton in which all execution sequences are finite, like R from Fig. 1, the probability of an action's occurrence is easily obtained: we calculate probabilities for the maximal execution sequences, given for each by multiplication of the probabilities of the choices occurring along it; the probability of an action's occurrence is then the sum of the so-calculated probabilities for

[5] We use this abstract definition because of its useful mathematical properties; it is equivalent to Segala's [17] in that it generates the same distributions over action sequences, as the following discussion illustrates.

[6] In this regard our resolutions strictly generalise the ones of Jonsson *et al.* [8].

We show that M is resolved by R in the following diagram. Enclosing circles and ovals represent distributions, the enclosed shapes' relative sizes within hinting at probabilities; on the left the shapes are associated 1-1 with the distinct states, but the right-hand states' shapes indicate the left-hand states *to which they are mapped*. Thus the resolving function f is given by $f.r_{1,2,3,4} := m_1$, $f.r_5 := m_2$ and $f.r_6 := m_3$.



The left-hand automaton M has initial (Dirac) distribution $m^\circ = \overline{m_1}$ and transitions

$$\mathcal{T}.m_1 \;:=\; \{(\alpha, \overline{m_1})\ ,\ (\beta, \overline{m_2} \;{}_{\frac{1}{2}}\!\oplus\, \overline{m_3})\ ,\ (\beta, \overline{m_3})\}\ , \qquad \mathcal{T}.m_2 \;=\; \mathcal{T}.m_3 \;:=\; \emptyset\ .$$

The right-hand (fully probabilistic) resolution R has initial distribution $r^\circ = \overline{r_1} \;{}_{\frac{1}{3}}\!\oplus\, \overline{r_2}$; its non-$\emptyset$ transitions are

$$\begin{array}{lcllcl} \mathcal{T}'.r_1 & := & \{(\alpha, \overline{r_3} \;{}_{\frac{1}{2}}\!\oplus\, \overline{r_4})\} & \mathcal{T}'.r_3 & := & \{(\alpha, \overline{r_2} \;{}_{\frac{1}{2}}\!\oplus\, \overline{r_4})\} \\ \mathcal{T}'.r_2 & := & \{(\beta, \overline{r_5} \;{}_{\frac{1}{2}}\!\oplus\, \overline{r_6})\} & \mathcal{T}'.r_4 & := & \{(\beta, \overline{r_6})\}\ . \end{array}$$

Fig. 1. Example of a fully probabilistic resolution

all maximal sequences containing it. For example, in R the maximal-execution-sequence probabilities are

$$\begin{array}{lll} \langle r_1, \alpha, r_3, \alpha, r_2, \beta, r_5\rangle & @\,(1/3{\times}1/2{\times}1/2{\times}1/2) & \text{that is, probability } 1/24 \\ \langle r_1, \alpha, r_3, \alpha, r_2, \beta, r_6\rangle & @\,(1/3{\times}1/2{\times}1/2{\times}1/2) & \text{probability } 1/24 \\ \langle r_1, \alpha, r_3, \alpha, r_4, \beta, r_6\rangle & @\,(1/3{\times}1/2{\times}1/2{\times}1) & \text{probability } 1/12 \\ \langle r_1, \alpha, r_4, \beta, r_6\rangle & @\,(1/3{\times}1/2{\times}1) & \text{probability } 1/6 \\ \langle r_2, \beta, r_5\rangle & @\,2/3{\times}1/2 & \text{probability } 1/3 \\ \langle r_2, \beta, r_6\rangle & @\,2/3{\times}1/2 & \text{probability } 1/3\ , \end{array}$$

with all other sequences being assigned probability zero. If we use f to map this distribution back to M, and aggregate, we get

$$\begin{array}{ll} \langle m_1, \alpha, m_1, \alpha, m_1, \beta, m_2\rangle & \text{probability } 1/24 \\ \langle m_1, \alpha, m_1, \alpha, m_1, \beta, m_3\rangle & \text{probability } 1/24 + 1/12 \\ \langle m_1, \alpha, m_1, \beta, m_3\rangle & \text{probability } 1/6 \\ \langle m_1, \beta, m_2\rangle & \text{probability } 1/3 \\ \langle m_1, \beta, m_3\rangle & \text{probability } 1/3\ , \end{array}$$

where we can see both history-dependence and interpolation at work. Finally, concentrating on just the actions gives us

$$\begin{array}{ll} \langle \alpha, \alpha, \beta \rangle & \text{probability } 1/24 + 1/24 + 1/12 \\ \langle \alpha, \beta \rangle & \text{probability } 1/6 \\ \langle \beta \rangle & \text{probability } 1/3 + 1/3 \quad . \end{array}$$

No matter which of the three views above we use, the probability that a particular action occurs is the probability assigned to the *set* of all sequences containing that action: thus the probability of α's occurrence is $1/24+1/24+1/12+1/6 = 1/3$; for β the probability is 1. (The sum exceeds one because both can occur.)

When our automata can generate infinite executions, however, there might be uncountably many of them: consider a looping automaton, choosing forever between bits 0 and 1, whose infinite sequences thus encode the whole unit interval $[0, 1]$ in binary. Simple summation within discrete distributions is then not appropriate;[7] in this case we apply the following more general definition.

Definition 3. Given a fully probabilistic automaton $\mathrm{R} = (R, r^\circ, E, I, \mathcal{T})$, the *probability that* R *starts with* a sequence of actions $\sigma \in \Sigma^*$, with $\Sigma := E \cup I$, is given by $r^\circ.(\mathrm{Pr}_{\mathrm{R}}.\sigma)$, where $\mathrm{Pr}_{\mathrm{R}} \in \Sigma^* \to R \to [0, 1]$ is defined inductively:

$$\mathrm{Pr}_{\mathrm{R}}.\varepsilon.r := 1 \quad \text{and} \quad \mathrm{Pr}_{\mathrm{R}}.(\alpha\sigma).r := \begin{cases} \mu.(\mathrm{Pr}_{\mathrm{R}}.\sigma) & \text{if } \mathcal{T}.r = \{(\alpha, \mu)\} \text{ for some } \mu \\ 0 & \text{otherwise} \end{cases}$$

Here ε denotes the empty sequence of actions and $\alpha\sigma$ the sequence starting with $\alpha \in \Sigma$ and continuing with $\sigma \in \Sigma^*$. Recall from Sec. 2.1 that $\mu.(\mathrm{Pr}_{\mathrm{R}}.\sigma)$ is the expected value over μ of the random variable $\mathrm{Pr}_{\mathrm{R}}.\sigma \in R \to [0, 1]$. The value $\mathrm{Pr}_{\mathrm{R}}.\sigma.r$ is the probability that R proceeds with sequence σ from state r.

Let $\Sigma^{*\alpha}$ be the set of finite sequences in Σ^* that contain α just once, namely at the end. Then the probability that a fully probabilistic automaton R ever performs an action α is given by $\sum_{\sigma \in \Sigma^{*\alpha}} r^\circ.(\mathrm{Pr}_{\mathrm{R}}.\sigma)$. [8] ¶

2.4 Probabilistic Testing

We now recall the testing framework of Segala [17] which, as we will see, differs from the testing framework of Wang and Larsen [21] in that a test may have countably many success actions rather than just one.[9]

---

[7] Discrete distributions' supports must be countable and so they cannot, for example, describe a situation in which the uncountably many infinite sequences are equally likely — unless they are all (equally) impossible.

[8] An alternative, but equivalent definition appeals to more general probabilistic measures: the probability of R's performing α is the measure of the set of sequences containing α at any point [17]. We have specialised here for simplicity.

[9] Another difference is that Segala's success actions must actually occur, whereas for Wang and Larsen (and earlier De Nicola and Hennessy [4]), it is sufficient that a state be reached from which the action is possible. Here we treat only the former.

We begin by defining the parallel composition of two automata in the CSP style [7], synchronising them on their common external actions.

Definition 4. Let $\mathrm{M}_1 = (M_1, m_1^\circ, E_1, I_1, \mathcal{T}_1)$ and $\mathrm{M}_2 = (M_2, m_2^\circ, E_2, I_2, \mathcal{T}_2)$ be two automata, and let $\Sigma_i := E_i \cup I_i$. They are *compatible* when the only actions they share are external, that is when $\Sigma_1 \cap \Sigma_2 \subseteq E_1 \cap E_2$. In that case their *parallel composition* $\mathrm{M}_1 \| \mathrm{M}_2$ is $(M_1 \times M_2, m_1^\circ \times m_2^\circ, E_1 \cup E_2, I_1 \cup I_2, \mathcal{T})$ where

$$\mathcal{T}.(m_1, m_2) := \{(\alpha,\, \mu_1 \times \mu_2) \mid \alpha \in \Sigma_1 \cup \Sigma_2 \quad \wedge \\ (\alpha, \mu_i) \in \mathcal{T}_i.m_i \text{ if } \alpha \in \Sigma_i \text{ else } \mu_i = \overline{m_i}, \text{for } i = 1, 2\}. \qquad \P$$

Parallel composition is the basis of testing: it models the interaction of the observer with the process being tested; and it models the observer himself — as an automaton.

From here on, we fix some disjoint sets $\mathbb{E}$ and $\mathbb{I}$ of external and internal actions, and the automata we subject to testing —the ones for which testing preorders will be defined— are the ones whose components of external- and internal actions are subsets of $\mathbb{E}$ and $\mathbb{I}$. Now let $\Omega := \{\omega_1, \omega_2, \cdots\}$ be a countable set of *success actions*, disjoint from $\mathbb{E}$ and $\mathbb{I}$, and define a *test* to be an automaton $\mathrm{T} = (T, t^\circ, E, I, \mathcal{T})$ with $E \supseteq \mathbb{E} \cup \Omega$ and $I \cap \mathbb{I} = \emptyset$. Thus every such test T is automatically compatible with the automaton M that is to be tested, and in the parallel composition $\mathrm{M} \| \mathrm{T}$ *all* the external actions of M must synchronise with actions of T. Let $\mathbb{T}$ be the class of all such tests, and write $\mathbb{T}_N$ for the subclass of $\mathbb{T}$ that uses only N success actions; we write $\mathbb{T}_*$ for $\bigcup_{N \in \mathbb{N}} \mathbb{T}_N$ and, for convenience, allow $\mathbb{T}_{\mathbb{N}}$ as a synonym for $\mathbb{T}$ itself.

To *apply* test T to automaton M we first form the composition $\mathrm{M} \| \mathrm{T}$ and then consider all resolutions of that composition separately: in each one, any particular success action ω_i will have some probability of occurring; and those probabilities, taken together, give us a single *success tuple* for the whole resolution, so that if w is the tuple then w_i is the recorded probability of ω_i's occurrence. The set of all those tuples, *i.e.* over all resolutions of $\mathrm{M} \| \mathrm{T}$, is then the complete outcome of applying test T to automaton M: as such, it will be a subset of $\mathbb{W} := [0,1]^{\mathbb{N}^+}$.

Definition 5. For a fully probabilistic automaton R, let its (single) success tuple $\mathbb{W}.\mathrm{R} \in [0,1]^{\mathbb{N}^+}$ be such that $(\mathbb{W}.\mathrm{R})_i$ is the probability that R performs the action ω_i, as given in Def. 3.

Then for a (not necessarily fully probabilistic) automaton M we define the set of its success tuples to be those resulting as above from all its resolutions:

$$\mathbb{W}.\mathrm{M} \quad := \quad \{\mathbb{W}.\mathrm{R} \mid \mathrm{R} \text{ is a resolution of M}\}. \qquad \P$$

We note that the success-tuple set $\mathbb{W}.\mathrm{M}$ is *convex* in the following sense:

Lemma 1. For any two tuples $w_1, w_2 \in \mathbb{W}.\mathrm{M}$, their weighted average $w_1 \,{}_p\!\oplus\, w_2$ is also in $\mathbb{W}.\mathrm{M}$ for any $p \in [0,1]$.

Proof. Let $\mathrm{R}_1, \mathrm{R}_2$ be the resolutions of M that gave rise to w_1, w_2. Form R as their disjoint union, except initially where we define $r^\circ := r_1^\circ \,{}_p\!\oplus\, r_2^\circ$. The new resolution R generates the interpolated tuple $w_1 \,{}_p\!\oplus\, w_2$ as required. $\P$

2.5 May- and Must Preorders

We now define various preorders $\sqsubseteq$ on testable automata; in general $\mathrm{M}_1 \sqsubseteq \mathrm{M}_2$ will mean that M_2 scores at least as well as M_1 does on certain tests.

For $w, w' \in \mathbb{W}$, we write $w \leq w'$ if $w_i \leq w'_i$ for all $i \in \mathbb{N}^+$. Given that Ω comprises "success" actions it is natural to regard $\leq$ on $\mathbb{W}$ as a (non-strict) "better than" order, *i.e.* that it is better to have higher probabilities for the occurrence of success actions. Since nondeterminism generates however *sets* of success tuples (Def. 5), rather than merely individuals, we are led to appeal to two complementary testing preorders on automata; they are based on the standard techniques for promoting an underlying order to a preorder on powersets.

Definition 6. Given two automata $\mathrm{M}_1, \mathrm{M}_2$ and a testing automaton T compatible with both, say that

$$\begin{array}{rcl} \mathrm{M}_1 \sqsubseteq^{\mathrm{T}}_{\mathtt{may}} \mathrm{M}_2 & \text{iff} & \mathbb{W}.(\mathrm{M}_1||\mathrm{T}) \leq_H \mathbb{W}.(\mathrm{M}_2||\mathrm{T}) \\ \mathrm{M}_1 \sqsubseteq^{\mathrm{T}}_{\mathtt{must}} \mathrm{M}_2 & \text{iff} & \mathbb{W}.(\mathrm{M}_1||\mathrm{T}) \leq_S \mathbb{W}.(\mathrm{M}_2||\mathrm{T}) \ , \end{array}$$

where $\leq_H, \leq_S$ are the Hoare, resp. Smyth preorders on $\mathcal{P}\mathbb{W}$ generated from the index-wise order $\leq$ on $\mathbb{W}$ itself.[10] Abstracting over all tests in $\mathbb{T}$ then gives us

$$\mathrm{M}_1 \sqsubseteq^{\mathbb{T}}_{_} \mathrm{M}_2 \qquad \text{iff} \qquad \forall\, \mathrm{T} \in \mathbb{T} : \mathrm{M}_1 \sqsubseteq^{\mathrm{T}}_{_} \mathrm{M}_2 \ ,$$

where within a single formula or phrase we use "$_$" for "`may`, `must` respectively" in the obvious way, and we define the preorders $\sqsubseteq^{\mathbb{T}}_{_}*$, and $\sqsubseteq^{\mathbb{T}_N}_{_}$ for $N \geq 1$, by similar abstractions. Finally, *scalar testing* as employed by Wang & Larsen [21] is defined by taking suprema and infima, as follows:

$$\begin{array}{rcl} \mathrm{M}_1 \sqsubseteq^{1}_{\mathtt{may}} \mathrm{M}_2 & \text{iff} & \forall\, \mathrm{T} \in \mathbb{T}_1 : \bigsqcup \mathbb{W}.(\mathrm{M}_1||\mathrm{T}) \leq \bigsqcup \mathbb{W}.(\mathrm{M}_2||\mathrm{T}) \\ \mathrm{M}_1 \sqsubseteq^{1}_{\mathtt{must}} \mathrm{M}_2 & \text{iff} & \forall\, \mathrm{T} \in \mathbb{T}_1 : \bigsqcap \mathbb{W}.(\mathrm{M}_1||\mathrm{T}) \leq \bigsqcap \mathbb{W}.(\mathrm{M}_2||\mathrm{T}) \ . \end{array} \qquad \P$$

Thus, in vector-based testing $\sqsubseteq^{\mathbb{T}}_{_}$, for each test one compares sets of *tuples* of reals, whereas for $\sqsubseteq^{\mathbb{T}_1}_{_}$ one is merely comparing sets of (mono-tuple-, hence scalar) reals. Then, by taking extrema, scalar testing abstracts even further to a comparison of single scalars.

Clearly $(\sqsubseteq^{\mathbb{T}}_{_}) \Rightarrow (\sqsubseteq^{\mathbb{T}}_{_}*) \Rightarrow (\sqsubseteq^{\mathbb{T}_1}_{_}) \Rightarrow (\sqsubseteq^{1}_{_})$. Our principal interest is in determining the situations in which the last is as powerful as all the others, that is when the reverse implications also hold, giving equivalence: we ask

> Under what conditions are *scalar* tests on their own sufficient to distinguish probabilistic automata?

In Sec. 5 we identify general criteria which suffice for scalar testing; then in Secs. 6 and 7 we identify classes of automata on which we can achieve those criteria. Sections 3 and 4 introduce "reward testing" for that purpose.

[10] The Hoare order is defined by $X \leq_H Y$ iff $\forall x \in X : \exists y \in Y : x \leq y$; similarly the Smyth order is defined by $X \leq_S Y$ iff $\forall y \in Y : \exists x \in X : x \leq y$ [1].

3 Reward Testing of Finite-Dimensional Tuple Sets

Def. 5 gives tuple sets $\mathbb{W}.(\mathrm{M}||\mathrm{T})$, coming from a testee M and test $\mathrm{T}\in\mathbb{T}_{\mathbb{N}}$; when in fact $\mathrm{T}\in\mathbb{T}_*$ they can be considered to lie in some N-dimensional unit cube $[0,1]^N$. We now abstract from automata, considering just those tuple sets; let N be fixed. This section shows it sufficient, under suitable conditions, to record only an "extremal expected reward" for each set of N-tuple outcomes, a single scalar rather than the whole tuple-set.

Definition 7. A *reward tuple* is an N-tuple $h\in[0,1]^N$ of real numbers; given such an h we define two forms of *extremal reward outcomes* with respect to any tuple set $W\subseteq[0,1]^N$:

- The *Hoare* outcome $h_H{\cdot}W$ is the supremum $\bigsqcup_{w\in W}(h\cdot w)$, and
- The *Smyth* outcome $h_S{\cdot}W$ is the infimum $\bigsqcap_{w\in W}(h\cdot w)$,

where within the extrema we write $h\cdot w$ for the dot-product of the two tuples.

Given a reward-tuple $h\in[0,1]^N$, the two forms of outcome give us two corresponding *reward orders* on tuple-sets $W_1,W_2\subseteq[0,1]^N$:

$$\begin{array}{rcl} W_1\leq^h_{\mathtt{may}} W_2 & \text{iff} & h_H{\cdot}W_1\leq h_H{\cdot}W_2 \\ W_1\leq^h_{\mathtt{must}} W_2 & \text{iff} & h_S{\cdot}W_1\leq h_S{\cdot}W_2\ , \end{array}$$

where (note) the comparison $\leq$ on the right has now been reduced to simple scalars. As in Def. 6, this generalises so that we can define

$$W_1\leq^N_{-} W_2 \qquad \text{iff} \qquad \forall h\in[0,1]^N : W_1\leq^h_{-} W_2\ . \qquad \P$$

We will now show that these preorders coincide with $\leq_H$ and $\leq_S$, respectively, provided the tuple-sets have a certain form of closure, as follows:

Definition 8. We say that a subset W of the N-dimensional Euclidean space is *p-closed* (for *probabilistically* closed) iff

- It is *convex*, that is if $w_1,w_2\in W$ and $p\in[0,1]$ then the weighted average $w_1\,{}_p\!\oplus\, w_2$ is also in W, and
- It is *Cauchy closed*, that is it contains all its limit points in the usual Euclidean metric, and it is *bounded*.[11] $\P$

Our sets' being *p-closed* will allow us to appeal to the *Separating Hyperplane Lemma* from discrete geometry [11, Thm. 1.2.4 paraphrased]:[12]

Lemma 2. Let A and B be two convex- and Cauchy-closed subsets of Euclidean N-space; assume that they are disjoint and that at least one of them is bounded. Then there is a hyperplane that strictly separates them. $\P$

Here a *hyperplane* is a set of the form $\{w\in\mathbb{R}^N \mid h{\cdot}w=c\}$ for certain $h\in\mathbb{R}^N$ (the *normal* of the hyperplane) and $c\in\mathbb{R}$, and such a hyperplane *strictly separates* A and B if for all $a\in A$ and $b\in B$ we have $h\cdot a<c<h\cdot b$ or $h\cdot a>c>h\cdot b$.

[11] Cauchy closure and boundedness together amounts to *compactness*.

[12] The hyperplanes are motivated indirectly by a proof of McIver [14, Lem. 8.2].

Our main theorem is then a direct application of Lem. 2: the normal h of the asserted hyperplane provides the rewards used in Def. 7.

Theorem 1. Let A, B be subsets of $[0,1]^N$; then we have

$$\begin{array}{lll} A \leq_H B & \text{iff} & A \leq^N_{\texttt{may}} B \qquad \text{if } B \text{ is } p\text{-}closed, \text{ and} \\ A \leq_S B & \text{iff} & A \leq^N_{\texttt{must}} B \qquad \text{if } A \text{ is } p\text{-}closed. \end{array}$$

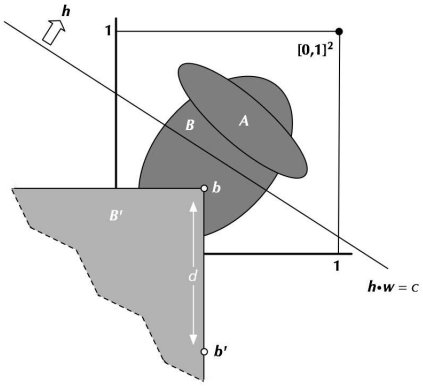
Proof. We consider first the *only-if* -direction for the Smyth/must case:

$$\begin{array}{lll} & A \leq_S B & \\ \Leftrightarrow & \forall b \in B : \exists a \in A : a \leq b & \text{defn. } \leq_S \\ \Rightarrow & \forall h \in [0,1]^N : \forall b \in B : \exists a \in A : h \cdot a \leq h \cdot b & h \geq 0 \\ \Rightarrow & \forall h \in [0,1]^N : \forall b \in B : h_S{\cdot}A \leq h \cdot b & h_S{\cdot}A \leq h \cdot a \\ \Leftrightarrow & \forall h \in [0,1]^N : h_S{\cdot}A \leq h_S{\cdot}B & \text{defn. } (h_S\cdot)\text{; properties of } \sqcap \\ \Leftrightarrow & \forall h \in [0,1]^N : A \leq^h_{\texttt{must}} B \ . & \text{defn. } \leq^h_{\texttt{must}} \\ \Leftrightarrow & A \leq^N_{\texttt{must}} B \ . & \text{defn. } \leq^N_{\texttt{must}} \end{array}$$

For the *if*-direction we use separating hyperplanes, proving the contrapositive:

$$\begin{array}{lll} & A \not\leq_S B & \\ \Leftrightarrow & \forall a \in A : \neg(a \leq b) & \text{defn. } \leq_S\text{; for some } b \in B \\ \Leftrightarrow & A \cap B' = \emptyset & \text{define } B' := \{b' \in \mathbb{R}^N \mid b' \leq b\} \end{array}$$

$$\Leftrightarrow \quad \exists h \in \mathbb{R}^N, c \in \mathbb{R} : \quad \text{Lem. 2; } A \text{ is } p\text{-}closed\text{; } B' \text{ is convex and Cauchy-closed}$$
$$\forall a \in A, b' \in B' : \\ h \cdot b' < c < h \cdot a \ ,$$



where *wlog* the inequality can be in the direction shown, else we simply multiply h, c by -1.

We now argue that h is non-negative, whence by scaling of h, c we obtain *wlog* that $h \in [0,1]^N$. Assume for a contradiction that $h_n < 0$. Choose scalar $d \geq 0$ large enough so that the point $b' := (b_1, \cdots, b_n - d, \cdots, b_N)$ falsifies $h \cdot b' < c$; since b' is still in B', however, that contradicts the separation. Thus we continue

$$\Leftrightarrow \quad \exists h \in [0,1]^N, c \in \mathbb{R} : \quad \text{above comments concerning } d$$
$$\forall a \in A, b' \in B' : \\ h \cdot b' < c < h \cdot a \ ,$$

$$\begin{array}{lll} \Leftrightarrow & \exists h \in [0,1]^N, c \in \mathbb{R} : \forall a \in A : h \cdot b < c < h \cdot a & \text{set } b' \text{ to } b\text{; note } b \in B' \\ \Rightarrow & \exists h \in [0,1]^N, c \in \mathbb{R} : h \cdot b < c \leq h_S{\cdot}A & \text{defn. } (h_S\cdot)\text{; properties of } \sqcap \\ \Rightarrow & \exists h \in [0,1]^N, c \in \mathbb{R} : h_S{\cdot}B < c \leq h_S{\cdot}A & b \in B\text{, hence } h_S{\cdot}B \leq h \cdot b \\ \Leftrightarrow & \exists h \in [0,1]^N : A \not\leq^h_{\texttt{must}} B & \text{defn. } \leq^h_{\texttt{must}} \\ \Leftrightarrow & A \not\leq^N_{\texttt{must}} B \ . & \text{defn. } \leq^N_{\texttt{must}} \end{array}$$

The proof for the Hoare-case is analogous. ¶

4 Reward Testing of Automata

We now combine Secs. 2 and 3 by introducing preorders that use testing processes and rewards together: we define

$$\mathrm{M}_1 \sqsubseteq^N_{-} \mathrm{M}_2 \qquad \text{iff} \qquad \forall\, \mathrm{T} \in \mathbb{T}_N : \mathbb{W}.(\mathrm{M}_1 \| \mathrm{T}) \leq^N_{-} \mathbb{W}.(\mathrm{M}_2 \| \mathrm{T})\ .$$

We call the $\sqsubseteq^N_{-}$ relations —and $\sqsubseteq^*_{-}$, $\sqsubseteq^{\mathbb{N}}_{-}$ similarly— the *reward-testing* preorders for automata. When N=1 this definition of $\sqsubseteq^1_{-}$ is just scalar testing again (from Sec. 2.5) as, in this case, the reward "vectors" are just scalars themselves: thus the notations do not clash. The following is an immediate corollary of Thm. 1.

Corollary 1. If M_1 and M_2 are automata with the property that for any test $\mathrm{T} \in \mathbb{T}_*$ the sets $\mathbb{W}.(\mathrm{M}_1\|\mathrm{T})$ and $\mathbb{W}.(\mathrm{M}_2\|\mathrm{T})$ are *p-closed*, then finite-dimensional vector-based testing is equivalent to reward testing: from Thm. 1 we have

$$\mathrm{M}_1 \sqsubseteq^{\mathbb{T}_N}_{-} \mathrm{M}_2 \quad \text{iff} \quad \mathrm{M}_1 \sqsubseteq^N_{-} \mathrm{M}_2 \qquad \text{for all } N,$$

which in turn implies that $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}*}_{-} \mathrm{M}_2$ iff $\mathrm{M}_1 \sqsubseteq^*_{-} \mathrm{M}_2$. ¶

5 Closure Suffices for Scalar Testing

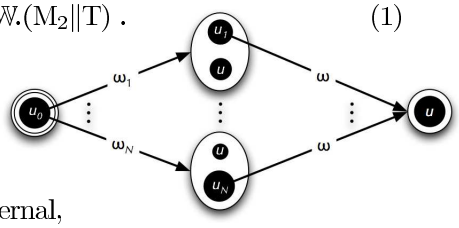
We now show that scalar testing is equally powerful as finite-dimensional reward testing which, with Cor. 1, implies that *p-closure* of the generated tuple-sets $\mathbb{W}.(\mathrm{M}\|\mathrm{T})$ is a sufficient condition for scalar testing to be as powerful as finite-dimensional vector-based testing. In doing so, we assume that tests are ω-terminal in the sense that they halt after execution of any success action.[13]

Theorem 2. For automata M_1 and M_2 we have that $\mathrm{M}_1 \sqsubseteq^*_{-} \mathrm{M}_2$ iff $\mathrm{M}_1 \sqsubseteq^1_{-} \mathrm{M}_2$.

Proof. The *only-if* is trivial in both cases. For *if* we prove the must-case in the contrapositive; the may-case is similar.

Suppose thus that $\mathrm{M}_1 \not\sqsubseteq^N_{\mathtt{must}} \mathrm{M}_2$, *i.e.* that $\mathrm{M}_1, \mathrm{M}_2$ are must-distinguished by some test $\mathrm{T} \in \mathbb{T}_N$ and reward $h \in [0,1]^N$, so that

$$\mathbb{W}.(\mathrm{M}_1\|\mathrm{T}) \quad \not\leq^h_{\mathtt{must}} \quad \mathbb{W}.(\mathrm{M}_2\|\mathrm{T})\ . \tag{1}$$



Assuming *wlog* that the success actions are $\omega_1, \cdots, \omega_N$ we construct an automaton U such that

- The state space is $\{u_0, \cdots, u_N\}$ and u,
- The actions are $\omega_1, \cdots \omega_N$ and ω, all external,
- The initial distribution u° is $\overline{u_0}$ and
- The transitions for $1 \leq i \leq N$ take u_0 via action ω_i to $(\overline{u_i}\ {}_{h_i}\!\oplus \overline{u})$, thence each u_i via ω to deadlock at $\overline{u}$.

[13] This assumption is justified in App. A.

We now consider the test T||U with ω as its only success action. In T||U an occurrence of ω_i is with probability h_i followed immediately by an occurrence of ω (and with probability $1-h_i$ by deadlock); and the overall probability of ω's occurrence, in any resolution of $\mathrm{M}_{1,2}||\mathrm{T}||\mathrm{U}$, is therefore the h-weighted reward $h \cdot w$ for the tuple $w := (w_1, \cdots, w_N)$ in the corresponding resolution of $\mathrm{M}_{1,2}||\mathrm{T}$.

Thus from (1) we have that $\mathrm{M}_1, \mathrm{M}_2$ can be distinguished using the scalar test T||U with its single success action ω; that is, we achieve $\mathrm{M}_1 \not\sqsubseteq^1_{\mathtt{must}} \mathrm{M}_2$ as required. ¶

6 Very Finite Testing is *p-closed*, hence Scalar

Say that an automaton is *very finite* if it is finite-state, finitely branching and loop-free (no execution sequence repeats a state), so that there is a finite upper bound on the length of its execution sequences.

In Thm. 3 below we show that scalar outcomes suffice when dealing with very finite automata and tests, because the sets of success tuples are *p-closed* and thus Thm. 2 applies. We rely on the fact that when a test T is very finite, so are the composite automata T||U and T||V constructed in Sec. 5 and App. A.

Lemma 3. Let $W_{1,\cdots,N} \subseteq \mathbb{W}$ be *p-closed* success-tuple sets, and let $\mu \in \overline{\{1,\cdots,N\}}$ be a discrete distribution over their indices. Then we have [14]

1. The set $\{\mu.f \mid f \in \{1,\cdots,N\} \to \mathbb{W} \ \wedge \ \forall i : f.i \in W_i\}$ is *p-closed*, and
2. The set $\bigcup_{\mu \in \overline{\{1,\cdots,N\}}} \{\mu.f \mid f \in \{1,\cdots,N\} \to \mathbb{W} \ \wedge \ \forall i : f.i \in W_i\}$ is *p-closed*.

That is, a specific interpolation μ of *p-closed* sets is again *p-closed* (1), and the union of *all* such interpolations is also *p-closed* (2).

Proof. Standard results on convex hulls of compact sets [2, Sec. 5.3]. ¶

Lemma 4. If M and T are very finite automata, then the set $\mathbb{W}.(\mathrm{M}||\mathrm{T})$ of success tuples is *p-closed*.

Proof. (Sketch) Lem. 3 shows that *p-closure* is preserved in a fairly straightforward induction on the upper bound of the length of the execution sequences of the very finite automata M||T. The details are omitted here, because Thm. 4 subsumes Thm. 3 anyway. ¶

Theorem 3. For very finite tests and automata, scalar testing suffices: we have

$$\mathrm{M}_1 \sqsubseteq^{\mathbb{T}_{\mathbb{N}}} \mathrm{M}_2 \quad \text{iff} \quad \mathrm{M}_1 \sqsubseteq^1 \mathrm{M}_2 \ .$$

Proof. Any very finite test has only finitely many success actions, and thus belongs to $\mathbb{T}_*$. Consequently, we have $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}_{\mathbb{N}}} \mathrm{M}_2$ iff $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}_*} \mathrm{M}_2$. Using this, the result follows from Lem. 4, Cor. 1 and Thm. 2. ¶

[14] Here we are taking the expected value of vectors: recall Footnote 4.

7 Also Finitary Testing is Scalar

We now remove the no-looping restriction of Sec. 6, retaining however that the automata are finite-branching and have finitely many states:[15] this allows their execution sequences to become of infinite length. Such automata we call *finitary*.

The result depends on a connection with Markov Decision Processes (*MDP*'s) [16], abstracted here as a lemma implicitly using the more general probability measures mentioned in Def. 3 (Footnote 8).

Lemma 5. *Static resolutions suffice for finitary testing* Say that a resolution R of an automaton M is *static* if its associated resolving function $f \in R \to M$ is injective, so that on every visit to a state $m \in M$ any nondeterminism is resolved in the same way, and does not interpolate. Then, for all reward tuples $h \in [0,1]^N$,

- There is a static resolution R_h of M so that $h_S \cdot (\mathbb{W}.\mathrm{M}) = h_S \cdot (\mathbb{W}.\mathrm{R}_h)$ and
- There is a static resolution R'_h of M so that $h_H \cdot (\mathbb{W}.\mathrm{M}) = h_H \cdot (\mathbb{W}.\mathrm{R}'_h)$.

Thus in both cases the extremum over all resolutions is attained statically.[16]

Proof. An automaton $\mathrm{M} = (M, m^\circ, E, I, \mathcal{T})$ with a reward tuple $h \in [0,1]^{\mathbb{N}^+}$ constitutes isomorphically an *MDP* [16, Sec. 2.1]: in the tuple at the end of Sec. 2.1.3 take $T := \mathbb{N}^+$, $S := M$, $A_s := \mathcal{T}.s$ for $s \in S$, $p_t(\cdot \mid s, (\alpha, \mu)) := \mu$ and $r_t(s, (\alpha, \mu)) := h_i$ if $\alpha = \omega_i$, or 0 if $\alpha \notin \Omega$. The values of p and r are independent of t and s. Our resolutions are the (history-dependent, randomised) policies of Sec. 2.1.5, and our Smyth- and Hoare-outcomes (Def. 7) are the optimal outcomes accruing from such policies [Secs. 2.1.6, 4.1 and 5.2]; the Smyth-case is obtained by using negative rewards so that "optimal" is supremum either way.

Theorem 7.1.9 [Sec. 7.1.4] states (paraphrased) *Suppose that the state space and the set of actions available at each state are finite. Then there is a stationary deterministic optimal policy.* "Stationary deterministic" equates to "static" in our setting, and "optimal" means "attains the extremum." ¶

The crucial lever that Lem. 5 gives us in our treatment of testing is that a finitary automaton has only finitely many static resolutions (up to isomorphism), since neither variation-by-visit nor interpolation is allowed for them. With infinitely many resolutions in general, even for the finitary case, needing only finitely many is a significant advantage — as we now see in the following lemmas.

Lemma 6. Let $\mathbb{W}_f.\mathrm{M}$ be the convex closure of the statically generated success tuples of M; then $\mathbb{W}_f.\mathrm{M} \subseteq \mathbb{W}.\mathrm{M}$. If M is finitary, then $\mathbb{W}_f.\mathrm{M}$ is *p-closed*.

Proof. The first statement is trivial, since static resolutions are still resolutions and from Lem. 1 we know that $\mathbb{W}.\mathrm{M}$ is convex. For the second we note that as

[15] Having finitely many states and transitions is an equivalent restriction. Finitely many states does not on its own imply finite branching, however: there are infinitely many distributions possible over even a finite state space.

[16] More general work on games [12] led us to this; a similar result is argued directly by Philippou, Lee & Sokolsky [15] and Cattani and Segala [3], in both cases in the context of decision procedures for bisimulation.

M has only finitely many static resolutions, the set $\mathbb{W}_f.\mathrm{M}$ is the convex-closure of a finite number of points, and is thus *p-closed* by Lem. 3(2). ¶

Lemma 7. For all finitary automata M with N success actions, and reward tuples $h \in [0,1]^N$, we have

$$h_S{\cdot}(\mathbb{W}.\mathrm{M}) = h_S{\cdot}(\mathbb{W}_f.\mathrm{M}) \qquad \text{and} \qquad h_H{\cdot}(\mathbb{W}.\mathrm{M}) = h_H{\cdot}(\mathbb{W}_f.\mathrm{M})\,,$$

hence $\mathbb{W}.\mathrm{M}$ and $\mathbb{W}_f.\mathrm{M}$ are equivalent under $\leq^N_{\underline{\ }}$.

Proof. The $\leq$-case for Smyth is immediate from Lem. 6; from Lem. 5 there is some static resolution R_h of M with $h_S{\cdot}(\mathbb{W}.\mathrm{M}) = h_S{\cdot}(\mathbb{W}.\mathrm{R}_h) \geq h_S{\cdot}(\mathbb{W}_f.\mathrm{M})$.

The Hoare-case is similar; the equivalence then follows from Def. 7. ¶

Lemmas 6 and 7 allow us to strengthen Cor. 1, effectively requiring *p-closure* only of $\mathbb{W}_f.(\mathrm{M}_{1,2}||\mathrm{T})$ rather than of $\mathbb{W}.(\mathrm{M}_{1,2}||\mathrm{T})$.

Lemma 8. For finitary automata $\mathrm{M}_1, \mathrm{M}_2$ we have $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}*}_{\underline{\ }} \mathrm{M}_2$ iff $\mathrm{M}_1 \sqsubseteq^{*}_{\underline{\ }} \mathrm{M}_2$.

Proof. For "only-if" apply Thm. 1 — this direction does not require *p*-closure. For *if* we prove the must-case in the contrapositive; the may-case is similar.

	$\mathrm{M}_1 \not\sqsubseteq^{\mathbb{T}_N}_{\mathtt{must}} \mathrm{M}_2$	for some N
$\Leftrightarrow$	$\mathbb{W}.(\mathrm{M}_1\|\|\mathrm{T}) \not\leq_S \mathbb{W}.(\mathrm{M}_2\|\|\mathrm{T})$	Def. 6, for some $\mathrm{T} \in \mathbb{T}_N$
$\Rightarrow$	$\mathbb{W}_f.(\mathrm{M}_1\|\|\mathrm{T}) \not\leq_S \mathbb{W}.(\mathrm{M}_2\|\|\mathrm{T})$	Lem. 6, $\mathbb{W}_f.(\mathrm{M}_1\|\|\mathrm{T}) \subseteq \mathbb{W}.(\mathrm{M}_1\|\|\mathrm{T})$
$\Leftrightarrow$	$\mathbb{W}_f.(\mathrm{M}_1\|\|\mathrm{T}) \not\leq^N_{\mathtt{must}} \mathbb{W}.(\mathrm{M}_2\|\|\mathrm{T})$	Lem. 6, $\mathbb{W}_f.(\mathrm{M}_1\|\|\mathrm{T})$ is *p-closed*; Thm. 1
$\Leftrightarrow$	$\mathbb{W}.(\mathrm{M}_1\|\|\mathrm{T}) \not\leq^N_{\mathtt{must}} \mathbb{W}.(\mathrm{M}_2\|\|\mathrm{T})$	Lem. 7
$\Rightarrow$	$\mathrm{M}_1 \not\sqsubseteq^N_{\mathtt{must}} \mathrm{M}_2$.	Definition of reward testing. ¶

We can now establish the extension of Thm. 3 to finitary automata.

Theorem 4. For finitary tests and automata, scalar testing suffices.

Proof. As in the proof of Thm. 3 we observe that $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}_{\mathbb{N}}}_{\underline{\ }} \mathrm{M}_2$ iff $\mathrm{M}_1 \sqsubseteq^{\mathbb{T}*}_{\underline{\ }} \mathrm{M}_2$ for (this time) finitary tests; we then apply Lem. 8 and Thm. 2. ¶

8 Beyond Finitary Testing

The principal technical ingredient of our results is *p-closure* of the result sets in $\mathbb{W}$, since that is what enables the hyperplane separation. Separation itself is not inherently finitary, since Lem. 2 extends to countably infinite dimensions [13, Lem. B.5.3 adapted], delivering a hyperplane whose normal is non-zero in only finitely many dimensions — just as required for our constructions above (although automaton V in App. A needs a slightly different approach).

It is the *p-closure* which (currently) depends essentially on finite dimensions (hence a finite state space). For countably infinite dimensions, its convexity component must be extended to infinite interpolations; but that happens automatically provided the sets are Cauchy closed. And it is again the closure that (within our bounded space) implies the compactness that separation requires. Thus Cauchy closure is the crucial property.

When the automata are finitely branching (up to interpolation), we believe a direct fixed-point construction of $\mathbb{W}.M$ (*i.e.* not indirectly via resolutions) gives, via intersection of a $\supseteq$-chain, a set whose up-closure (*wrt* $\leq$ over $\mathbb{W}$) is compact. Since must- (*i.e.* Smyth) testing is insensitive to up-closure, that would seem to extend our results to countably-infinite dimensional must-testing. For the may-testing case the analogous technique would be down-closure; but here, unfortunately, the limit is via union of a $\subseteq$-chain, and closure/compactness is not obviously preserved. As further evidence for this we note that the *MDP*-based results allow countably infinite state spaces in the infimum case (still however with finite branching) [16, Thm. 7.3.6], whereas the supremum case requires a finite state space.

Thus the key question is *What conditions are necessary to establish p-closure for infinitely many states?* At present, may- seems harder than must-.

9 Conclusion

Our reduction of vector- to scalar testing uses geometry (hyperplanes), elementary topology (compactness) and game theory (*MDP*'s); and it highlights the importance of *p-closure* and static resolutions. That those techniques contribute to probabilistic semantics is of independent theoretical interest; but our result ultimately is *practical* — any program calculus/algebra, a fundamental tool for rigorous software engineering, relies crucially on a tractable definition of equality.

A key feature is our use of *expected values of "rewards"* as outcomes for probabilistic tests; this approach subsumes the usual direct probabilities because an event's probability is just the expected value of its characteristic function.[17] It has been suggested before both for sequential [10] and concurrent [8] applications; and we believe it deserves greater attention, since it can lead to significant simplifications (of which this report provides an example).

We have shown that scalar testing suffices when requiring the *tests*, as well as their testees, to be finitary, and one may wonder whether the same holds if arbitrarily complex tests are allowed. We think this is indeed the case, for we conjecture that if two finitary automata can be distinguished by an arbitrary test, then they can be distinguished by a finitary test.

As stated in Sec. 2.4, Segala's testing framework [17] differs from others' [4, 21] not only in the number of success actions, but also in how to report success. We claim that action- and state-based testing give rise to different must-testing preorders when the tested processes display divergence, because a success action ω may be delayed forever. Here we used action-based testing throughout, although we could have obtained the same results for state-based testing.

Finally, we note that our reduction via Thms. 2–4 of vector-based testing to extremal scalar testing has reduction to $\mathbb{T}_1$-testing as a corollary — thus showing that, under our assumptions, single success actions suffice for Segala's approach even if it is otherwise unchanged.

[17] For $\mu.X$ is the same whether X is taken as a set or as a characteristic function.

Acknowledgements

We thank Anna Philippou for corresponding with us about static resolutions, and Kim Border for a hyperplane-related counterexample.

References

1. S. Abramsky & A. Jung (1994): *Domain theory.* In S. Abramsky, D.M. Gabbay & T.S.E. Maibaum, editors: *Handbook of Logic and Computer Science*, volume 3, Clarendon Press, pp. 1–168.
2. C.D. Aliprantis & K.C. Border (1999): *Infinite Dimensional Analysis.* Springer, second edition.
3. S. Cattani & R. Segala (2002): *Decision algorithms for probabilistic bisimulation.* In Proc. *CONCUR 2002*, LNCS 2421, Springer, pp. 371–85.
4. R. De Nicola & M. Hennessy (1984): *Testing equivalences for processes.* *Theoretical Computer Science* 34, pp. 83–133.
5. H. Hansson & B. Jonsson (1990): *A calculus for communicating systems with time and probabilities.* In Proc. of the *Real-Time Systems Symposium* (RTSS '90), IEEE Computer Society Press, pp. 278–87.
6. He Jifeng, K. Seidel & A.K. McIver (1997): *Probabilistic models for the guarded command language.* *Science of Computer Programming* 28, pp. 171–92.
7. C.A.R. Hoare (1985): *Communicating Sequential Processes.* Prentice Hall.
8. B. Jonsson, C. Ho-Stuart & Wang Yi (1994): *Testing and refinement for nondeterministic and probabilistic processes.* In Proc. *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, Springer, pp. 418–30.
9. B. Jonsson & Wang Yi (2002): *Testing preorders for probabilistic processes can be characterized by simulations.* *Theoretical Computer Science* 282(1), pp. 33–51.
10. D. Kozen (1985): *A probabilistic PDL.* *Jnl. Comp. Sys. Sciences* 30(2), pp. 162–78.
11. J. Matoušek (2002): *Lectures on Discrete Geometry.* Springer.
12. A.K. McIver & C.C. Morgan (2002): *Games, probability and the quantitative μ-calculus qMu.* In *Proc. LPAR*, LNAI 2514, Springer, pp. 292–310.
13. A.K. McIver & C.C. Morgan (2005): *Abstraction, Refinement and Proof for Probabilistic Systems.* Tech. Mono. Comp. Sci., Springer.
14. C.C. Morgan, A.K. McIver & K. Seidel (1996): *Probabilistic predicate transformers.* *ACM Trans. on Programming Languages and Systems* 18(3), pp. 325–53.
15. A. Philippou, I. Lee & O. Sokolsky (2000): *Weak bisimulation for probabilistic systems.* In Proc. *CONCUR 2000*, Springer, pp. 334–49.
16. M.L. Puterman (1994): *Markov Decision Processes.* Wiley.
17. R. Segala (1996): *Testing probabilistic automata.* In Proc. *CONCUR '96*, LNCS 1119, Springer, pp. 299–314.
18. R. Segala & N.A. Lynch (1994): *Probabilistic simulations for probabilistic processes.* In Proc. *CONCUR '94*, LNCS 836, Springer, pp. 481–96.
19. M.I.A. Stoelinga & F.W. Vaandrager (2003): A testing scenario for probabilistic automata. In Proc. *ICALP '03*, LNCS 2719, Springer, pp. 407–18.
20. M.Y. Vardi (1985): *Automatic verification of probabilistic concurrent finite state programs.* In Proc. *FOCS '85*, IEEE Computer Society Press, pp. 327–38.
21. Wang Yi & K.G. Larsen (1992): *Testing probabilistic and nondeterministic processes.* In Proc. IFIP TC6/WG6.1 Twelfth Intern. Symp. on *Protocol Specification, Testing and Verification*, *IFIP Transactions* C-8, North-Holland, pp. 47–61.

A One Success Never Leads to Another

Here we substantiate the claim made in Sec. 6 (Footnote 13) that *wlog* we can assume that our testing automata halt after engaging in any success action. The reward-testing construction requires this because the automaton U used in Thm. 2 implementing a particular reward-tuple h effectively causes the composite automaton T||U to halt after the reward is applied — hence for correctness of the construction we required that the automaton T must have halted anyway.

Below we show that the may- and must testing preorders do not change upon restricting the class of available tests to those that cannot perform multiple success actions in a single run. A second reduction to tests that actually halt after performing any success action is trivial: just change any transition of the form (s, ω_i, μ) into a transition $(s, \omega_i, \overline{0})$ leading to a deadlocked state 0.

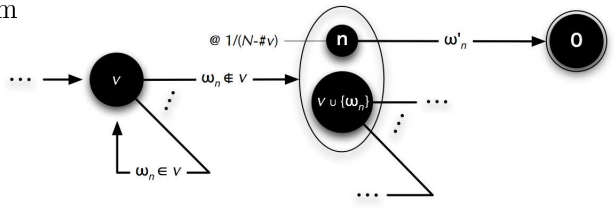
Suppose our original testing automaton T has N success actions $\omega_1, \cdots, \omega_N$. By running it in parallel with another automaton V (below) we will convert it to an automaton with the required property and corresponding success actions $\omega'_1, \cdots, \omega'_N$, and with success N-tuples that are exactly $1/N$ times the success tuples of T; since testing is insensitive to scaling of the tuples, that will give us our result. Note that the $1/N$ factor is natural given that we are making our N success actions mutually exclusive: it ensures the tuple's elements' total does not exceed one.

We construct the automaton $\mathrm{V} := (V, v^\circ, E, I, \mathcal{T})$ as follows:

- The state space is $V := \mathcal{P}\{\omega_1, \ldots, \omega_n\} \cup (0, \ldots, N)$, where the powerset-states record which success actions have already occurred, the scalar states $1, \ldots, N$ are "about-to-terminate" states, and 0 is a deadlock state;
- The actions are $\omega_1, \ldots, \omega_N$ and $\omega'_1, \ldots, \omega'_N$, all external; and
- The initial distribution v° is the Dirac'd powerset-state $\overline{\emptyset}$.

The transitions of V are of three kinds:

- "Terminating" transitions take state n with probability one via action ω'_n to the deadlocked state 0;
- "Do-nothing" transitions, from state $v \in \mathcal{P}\{\omega_1, \cdots, \omega_N\}$, lead with probability one via action $\omega_n \in v$ back to v, implementing that second and subsequent occurrences of any success action in T can be ignored; and
- "Success" transitions, from state $v \in \mathcal{P}\{\omega_1, \cdots, \omega_N\}$ lead via action $\omega_n \notin v$ with probability $\frac{1}{N - \#v}$ to state n, whence the subsequent terminating transition will emit ω'_n; the remaining probability at v leads to state $v \cup \{\omega_n\}$, recording silently that ω_n has now been taken care of.



When the original test T has finitely many states and transitions, so does the composite test T||V.