Unifying Probability with Nondeterminism

Yifeng Chen[1] and J. W. Sanders[2]

[1] HCST Key Lab at School of EECS, Peking University, China
[2] International Institute for Software Technology, UNU, Macao

Abstract. Early support for reasoning about probabilistic system behaviour replaced nondeterminism with probabilism. Only relatively recently have formalisms been studied that combine the two, and hence facilitate reasoning about probabilistic systems at levels of abstraction more general than code. Such studies have revealed an unsuspected subtlety in the interaction between nondeterministic and probabilistic choices that can be summarised: the demon resolving the nondeterministic choice has memory of previous state changes, whilst the probabilistic choice is made spontaneously. As a result, assignments to distinct variables need no longer commute. This paper introduces a model with explicit control of the length of the demon's memory. It does so by expanding the standard (initial-final) state view of computation to incorporate a third state, the 'original' state which checkpoints the most recent nondeterministic choice. That enables a nondeterministic choice to be made on the basis of only certain past probabilistic choices and so facilitates independent nondeterministic combinations to be chosen against just those. Sound laws are presented and used to analyse first an example in which no new behaviour should result, and second one that lies beyond the scope of traditional models.

1 Introduction

Models of sequential systems that account for both (demonic) nondeterministic and probabilistic choice are necessary in order to reason about system behaviour at levels of abstraction more general than code. They find use, for example, in the top-down incremental derivation of implementations that have been unpopular since Dijkstra introduced them in the *wp* setting. However the interaction between the two forms of choice turned out to be more subtle than was at first supposed. Here is a representataive example [4]: for programs A and B, let $A \sqcap B$ and $A \mathrel{{}_{\frac{1}{2}}\!\oplus} B$ denote their nondeterministic and fair probabilistic choices respectively; in the latter case, the result is equally likely to be A or B. Then, using standard notation for assignment,

$$\begin{array}{l}(x := 0 \sqcap x := 1) \mathbin{;} (y := 0 \mathrel{{}_{\frac{1}{2}}\!\oplus} y := 1) \neq \\ (y := 0 \mathrel{{}_{\frac{1}{2}}\!\oplus} y := 1) \mathbin{;} (x := 0 \sqcap x := 1)\,.\end{array} \tag{1}$$

Indeed, the probabilistic choice on the left is made *after* a value is assigned to x by the nondeterministic choice. Thus the probability of guaranteeing the

A. Cavalcanti and D. Dams (Eds.): FM 2009, LNCS 5850, pp. 467–482, 2009.

condition $x = y$ is $1/2$. But the right-hand program behaves more nondeterministically: the nondeterministic choice can observe, and so exploit, the preceding probabilistic choice, so that in the worst case it may keep ensuring $x \neq y$; thus the probability of guaranteeing $x = y$ is reduced to 0. In the presence of both probability and nondeterminism, disjoint assignments no longer commute!

Of course in standard sequential programming, disjoint assignments commute as a result of the two distributivity laws:

$$(B \sqcap C) \fatsemi A \;=\; (B \fatsemi A) \sqcap (C \fatsemi A)$$
$$A \fatsemi (B \sqcap C) \;=\; (A \fatsemi B) \sqcap (A \fatsemi C).$$

It is the second that fails in probabilistic programming. That feature of probabilistic systems has been used to good effect in modelling information flow (for example [1,9] will be discussed in Section 4). But 'unadulterated' nondeterminism has been found to be too strong, so that weaker versions have been considered. For instance, complementing [9], a restricted form of choice able to read only certain declared variables has been considered in [12]. Here we adopt a different approach and curtail the demons's memory. With little effort modifications are possible to make the demon prescient, if that is required, or to have more complex memory (see Section 5).

We consider probability, as has become standard, in the guise of a binary combinator $A \,{}_p\!\oplus\, B$ that chooses program A with probability p and B with the deficit probability $(1–p)$. That suffices to express known probabilistic algorithms and to reason about a moderately broad range of probabilistic behaviour [8]. It is shown there that Dijsktra's guarded-command language augmented with ${}_p\!\oplus$ provides a remarkably simple uniform notation facilitating the simultaneous treatment of functional and probabilistic behaviour. So with almost no extra complexity, probabilistic behaviour need not be handled by a 'second pass', but all at once with the consideration of input/output behaviour.

Two models of probability and nondeterminism have been developed in which the revised laws are sound: an expectation-transformer model [8] corresponding to the predicate-transformer model of standard programming, and a distributional model [8], corresponding to the standard binary-relational model. In the transformer model, each computation is modelled by transforming a random variable over final states (seen as an expectation) to the (pointwise) greatest expected value over initial states that can be guaranteed by executing the computation with 'reward' the post random variable. In the distributional model, each computation is seen as transforming an initial state to a set of distributions over final states (though in this paper we prefer to make it a relation between distributions). The distributional model is embedded in the transformer model by a Galois connection [8]. In the distributional model where nondeterminism is union, if sequential composition were to be the usual sequential composition of relations then equality would hold in (1). Thus the definition of sequential composition is more sophisticated in [4]. In common sequential programming, a relational composition $(S \fatsemi T)$ relates the final state of S and the initial state of T, reflecting the sequential state changes. The sequential composition of [4], however, also relates the initial state (distribution) of S with the final state of

T. Note this difference remains even when the state-distribution transformers are lifted to distribution transformers.

Here we follow the alternative of using the standard relational definition of sequential composition but a slightly more sophisticated definition of nondeterminism. We start from the observation that nondeterministic choices made after a sequential composition may be correlated with what happened well before. Thus we introduce a model of computation that enables a 'nondeterministic log' to be maintained and a 'nondeterministic checkpoint' to be taken to reset the log. The state from which the log extends is called the 'original' state of the computation; and we consider distributions over final state. A nondeterministic choice is resolved by reference to the log. But when the log is reset, original and current states coincide and there is no history for the demon to work from; the result is 'nondeterministically closed'. A computation then becomes a relation between such distributions. Three novel aspects of the model are:

1. each program construct is able to observe not only current state but also some original state, allowing sequential composition to be defined succinctly as a simple relational composition;
2. nondeterministic choice is separated into two operators: one binary operator $\oplus$ that arbitrarily combines probabilistic outcomes of two programs without the 'demonic' ability to act against previous probabilistic choices and another unary operator $\Box$ that performs the demonic act; such separation opens the door to include multiple nondeterministic choices with different backward-looking abilities;
3. the above two aspects explicitly reveal the interaction between probability and nondeterminism, and allow a (generalised) assignment to take into account observation of the original state, and 'angelically' act against a previous (demonic) nondeterministic choice, effectively achieving a kind of compensation.

In Section 2 the new model is described and sound laws presented. Proofs are largely routine using relational or predicate calculus. Section 3 contains a discussion of (1) to demonstrate that the new model preserves its properties; it then discusses the well-known Monty-Hall puzzle, and discusses a variant not able to be handled by previous models.

2 A Relational Model for Probabilistic Programming

In this section we introduce the notion of 'original state' and probability distributions that depend on original states. The semantic model supports seven basic commands. Healthiness conditions are introduced and sound algebraic laws are identified and used to transform programs into a normal form.

2.1 Distributions

Let $S = (V \to C)$ be the (finite) set of all states, each a mapping from program variables (denoted $x, y, \cdots, z$) to constants. Let $r, s, t, s_0, \cdots$ denote individual

states. A (conditional) *probability distribution* is a function: $h : S \to (S \to [0,1])$ whose first argument s denotes the *original state* and second argument denotes the *current state*. Let $\mathbb{H}$ denote the set of such distributions, with members $h, h_0, h_1, \cdots$. Distributions are partially ordered: $h_1 \leqslant h_2$ iff $h_1.s.t \leqslant h_2.s.t$ for all $s, t \in S$.

A *well-formed* distribution d satisfies: from any original state s, the total probability of all current states is at most 1: $\sum_t d.s.t \leqslant 1$. Let $\mathbb{D}$ denote the set of well-formed distributions, with members $d, d_0, d_1, \cdots$. A well-formed distribution is a probability distribution over current states conditional on a specific original state. A well-formed distribution satisfying $d.s.t = p$ records that if the *original state* is s, then the probability of *reaching this point* of the program in a *current state* t is p. Thus the final distribution of a nonterminating program may have total probability less than 1. A *functional distribution* f is a well-formed distribution and $f.s.t$ is either 0 or 1 for all s and t. For example, the delta distribution δ is functional and yields probability 1 when the current state equals the original state: $\delta.s.t = 1$ iff $s = t$. Let $\mathbb{F}$ denote the set of functional distributions. We have $\mathbb{F} \subseteq \mathbb{D} \subseteq \mathbb{H}$.

For program reasoning, we adopt convenient notation. Suppose $V = \{ x, y \}$; then $\langle x + y, x - y \rangle$ denotes a functional distribution indicating that the variables x and y have been updated to $x + y$ and $x - y$ respectively since the original state. The functional distribution $\langle x, y \rangle$, on the other hand, corresponds to the delta function in this context.

A *uniform distribution* u is a probability distribution such that for all t_1 and t_2 we have $u.s.t_1 = u.s.t_2$. Let $\mathbb{U}$ denote the set, with members $u, u_0, u_1, \cdots$. A *constant distribution* is uniform and for all s_1 and s_2, it satisfies $u.s_1.t = u.s_2.t$. Let $\mathbb{C}$ denote the set of constant distributions. For example, $1 \in \mathbb{C}$: for all states $1.s.t = 1$. We have $\mathbb{C} \subseteq \mathbb{U} \subseteq \mathbb{H}$. Symmetrically, a *current distribution* v is a probability distribution that is unchanged for every original state. Let $\mathbb{V}$ denote the set of current distributions, with members $v, v_0, \cdots$; then $\mathbb{C} \subseteq \mathbb{V} \subseteq \mathbb{H}$.

A *boolean distribution* b is such that for all s and t, the probability $b.s.t$ is either 0 or 1. Let $\mathbb{B}$ denote the set of boolean distributions, with members $b, b_0, b_1, \cdots$. Functional distributions are singleton boolean distributions which may depend on the original state. Evidently, $\mathbb{F} \subseteq \mathbb{B} \subseteq \mathbb{H}$.

We will use a notation $[S_0]$ where $S_0 \subseteq S$ to denote a boolean condition independent of the original state: $[S_0].s.t = 1 \;=\; (t \in S_0)$. For example, $[x = 1 \wedge y = 2]$ is equal to functional distribution $\langle 1, 2 \rangle$, and $[x = y]$ denotes the distribution such that $b.s.t = 1$ iff $t(x) = t(y)$ and $b.s.t = 0$ otherwise, and $[x = y] = \langle 0, 0 \rangle + \langle 1, 1 \rangle$. We have $[S_0] \in \mathbb{V}$.

2.2 Operations on Distributions

The *inner product* (over current states from every original state) between a well-formed distribution and a distribution is defined: $(d \cdot h).s.t \;\widehat{=}\; \sum_r d.s.r \times h.s.r$. The result distribution is always uniform. For example, the inner product $d \cdot [x = y]$ represents the total probability of the current states that satisfy $x = y$ from each original state.

The *linear combination* of two distributions h_1 and h_2 by a factor (uniform) distribution u is defined: $(h_1 \,{}_u\!\oplus\, h_2).s.t \;\widehat{=}\; h_1.s.t * u.s.t + h_2.s.t * (1 - u.s.t)$. This definition allows the choosing factors (probabilities) to depend on original states. The *convex combination* $(d_1 \,{}_u\!\oplus\, d_2)$ of two well-formed distributions by a uniform factor u is also well-formed. Inner product distributes over convex combination: $(d_1 \,{}_u\!\oplus\, d_2) \cdot h = (d_1 \cdot h) \,{}_u\!\oplus\, (d_2 \cdot h)$.

The *state update* of a well-formed distribution by a program expression $e : S \to (S \to S)$ is defined: $(d \dagger e).s.t \;\widehat{=}\; \sum_{r:e(s,r)=t} d.s.r$. From any original state (*i.e.* the first argument), the probability of a final state is the total probability of the initial states mapped into the final state. The result of an update is always well-formed. Assignments with program expressions that depend on the original state can be applied to perform backward compensation (see Section 3). The *current-state composition* $(e_1 \circ e_2)$ of two expressions is defined: $(e_1 \circ e_2)(s,t) \;\widehat{=}\; e_1(s, e_2(s,t))$ (see its use in Law 1(3)). Two consecutive state updates correspond to the current-state composition of the expressions: $(d \dagger e_1) \dagger e_2 = d \dagger (e_2 \circ e_1)$. A similar composition for distribution and program expression is defined: $(h \circ e).s.t \;\widehat{=}\; h.s.e(s,t)$ (see its use in Law 2(5)). State update distributes over convex combination: $(d_1 \,{}_u\!\oplus\, d_2) \dagger e = (d_1 \dagger e) \,{}_u\!\oplus\, (d_2 \dagger e)$. A useful equation relating inner product, state update and current-state composition is: $(d \dagger e) \cdot h = d \cdot (h \circ e)$.

The *convolution composition* between a well-formed distribution d and a distribution h is defined: $(d \otimes h).s.t \;\widehat{=}\; \sum_r d.s.r \times h.r.t$ (like matrix product it is associative). This composition represents the conditional probabilities in correspondence with the original states. It is easy to check that the result is always well-formed, and if h is uniform so is the result. The delta distribution δ is the unit of $\otimes$. In general, convolution does not distribute over probabilistic choice for well-formed distributions, although the distributivity does hold in special cases when the first argument of convolution is a functional distribution, the factor function on the right is a current distribution, or the factor function on the left is uniform:

$$\begin{array}{rcl} f \otimes (d_1 \,{}_h\!\oplus\, d_2) & = & (f \otimes d_1) \,{}_h\!\oplus\, (f \otimes d_2) \\ d \otimes (d_1 \,{}_v\!\oplus\, d_2) & = & (d \otimes d_1) \,{}_v\!\oplus\, (d \otimes d_2) \\ (d_1 \,{}_u\!\oplus\, d_2) \otimes d & = & (d_1 \otimes d) \,{}_u\!\oplus\, (d_2 \otimes d). \end{array}$$

2.3 The Semantic Model

A computation is represented by a relation $A(d, d')$ between an initial (well-formed) distribution d and a final (well-formed) distribution d'. The initial (or final) distribution represents the probability distribution over the initial (or final) states conditional on the probabilistic choice taken back in the original state. The basic commands are defined as follows.

Abort $\bot$, representing nontermination, is the most nondeterministic computation and may end up in any final distribution from every initial distribution. The *assignment* $t := e(s,t)$ uses a program expression e to modify the state according to the initial state t and the original state s. The probability of a final

state is the sum probabilities of the initial states mapped to it, all dependent on the original states. The final distribution may contain as much probability for nontermination as the initial distribution. *Skip* $\mathbb{I} = (t := t)$ (or alternatively $\mathbb{I} = (d' \geqslant d)$), is the unit of sequential composition. We adopt the view that if a computation, with a certain probability, does not terminate, then within that probability, its behaviour is chaotic: it may be in any state or may be in no state. Technically, the final distribution is always upwards-closed.

$\bot$	$\widehat{=}$	$true$
$t := e$	$\widehat{=}$	$d' \geqslant d \dagger e$
$A \fatsemi B$	$\widehat{=}$	$\exists d_0 \cdot A(d, d_0) \wedge B(d_0, d')$
$A \mathrel{{}_h\!\oplus} B$	$\widehat{=}$	$\exists d_1, d_2 \cdot A(d, d_1) \wedge B(d, d_2) \wedge d' \geqslant d_1 \mathrel{{}_{d \cdot h}\!\oplus} d_2$
$A \oplus B$	$\widehat{=}$	$\bigcup_h \; A \mathrel{{}_h\!\oplus} B$
$\Box(A)$	$\widehat{=}$	$\exists d_0 \cdot A(\delta,\, d_0) \;\wedge\; d' \geqslant d \otimes d_0$
μF	$\widehat{=}$	$\bigcup \{\, A \mid A \subseteq F(A) \,\}$

Sequential composition $A \fatsemi B$ equates, and hides, the final distribution of A and the initial distribution of B. *Probabilistic choice* $A \mathrel{{}_h\!\oplus} B$ with a choosing distribution h linearly combines the result distributions from A and B. *Open nondeterministic choice* $A \oplus B$ applies the universal union of convex combinations (with arbitrary choosing distributions) between relations. Note that finite union of relations, violating convexity closure, is not closed in the semantics, but open nondeterminism is. *Nondeterministic closure* $\Box(A)$ resets the initial state of A to be the original state; it is achieved with the delta distribution δ. The results from all possible initial states are recorded in d_0, representing how the final distributions depend on the initial states. The result distribution is convoluted with the initial distribution d to reflect the influence of the probability distribution of initial states. The definition of closure illustrates how a computation can take advantage of the history (as far back as the beginning of the nearest closure). *Recursion* is defined as the weakest (or largest) fixpoint μF where $F = F(X)$ is a program context that maps each relation X to another relation $F(X)$.

A derived command, the backward-looking nondeterministic choice $A \sqcap B \;\widehat{=}\; \Box(A \oplus B)$, corresponds to that of the standard probabilistic models and performs arbitrary convex combinations against the state at the beginning. Section 3.1 will show how this allows nondeterminism to act against the original probabilistic choices. *Binary conditional* that chooses A if b is true and otherwise B is a special case of probabilistic choice, equalling $A \mathrel{{}_b\!\oplus} B$ where b is a boolean distribution. Note that both assignment and binary conditional may depend on original states. Their combined uses can support more sophisticated forms of compensation (see Section 3.2).

2.4 Healthiness Conditions

Healthiness conditions can be viewed as imposed properties that yield desirable laws. The total probability (over initial states) from each original state is at

most 1. When it is less than 1, the deficiency represents the probability of nontermination. A principle of the unifying approach [5] and other totally-correct models is to assume that if the computation has not started then the computation becomes chaotic. In our probabilistic model, this is ensured by a healthiness condition: $A = (d{=}0 \vee A)$. Another healthiness condition ensures that if the computation may not terminate, the final distribution is chaotic for the probability of nontermination: $A = (A \fatsemi \text{Ш})$. Note that Ш itself also satisfies the first healthiness condition. Symmetrically, as a healthiness condition, skip is the left unit of sequential composition: $A = (\text{Ш} \fatsemi A)$. This ensures that for a specific final distribution, the possible initial distributions are downwards-closed. Arbitrary convex combinations of the final distributions (independently) from each initial state are closed (*i.e.* idempotence of $\oplus$): $A = (A \oplus A)$. This healthiness condition ensures idempotence of probabilistic choice in Law 2(1). In this paper, we assume that all computations are feasible (*i.e.* free of miracles) so that from any initial distribution, there exists some final distribution: $(A \fatsemi \bot) = \bot$. The fixpoint of our model uses Tarski's fixpoint theory and hence does not require the healthiness condition of Cauchy closure for continuity.

A specification A is called *nondeterministically closed* if $A = \Box(A)$. Such a computation does not depend on original states. An assignment $t{:=}e$ is closed if $e{=}e(t)$ does not depend on the original state s. A probabilistic choice $A \,{}_h\!\oplus\, B$ is closed if $h \in \mathbb{V}$ does not depend on the original state. Open nondeterminism $A \oplus B$ and sequential composition $A \fatsemi B$ are closed if both arguments are closed. Standard probabilistic programming corresponds to the sub-theory of nondeterministically closed specifications in the new model. Open specifications are useful if there exists some implementation mechanism (*e.g.* using a log file) that allows a computation to compensate against nondeterministic damage of undesirable errors in the past.

2.5 Algebraic Laws and Normal Form

The algebraic laws of this section are semantically sound. Law 1(1) and (2) are direct results of the healthiness conditions:

Law 1. (1) $\bot \fatsemi A = \bot = A \fatsemi \bot$ (2) $\text{Ш} \fatsemi A = A = A \fatsemi \text{Ш}$
(3) $t{:=}e_1 \fatsemi t{:=}e_2 = t{:=}e_2 \circ e_1$.

The following laws identify expected properties of probabilistic choice:

Law 2
(1) $A \,{}_h\!\oplus\, A = A$ (2) $A \,{}_h\!\oplus\, B = B \,{}_{1-h}\!\oplus\, A$
(3) $(A \,{}_h\!\oplus\, B) \,{}_{h'}\!\oplus\, C = A \,{}_{hh'}\!\oplus\, (B \,{}_{h''}\!\oplus\, C)$ *where* $h'' = h'(1-h)/(1-hh')$, $h < 1$
(4) $(A \,{}_h\!\oplus\, B) \fatsemi C = (A \fatsemi C) \,{}_h\!\oplus\, (B \fatsemi C)$
(5) $t{:=}e \fatsemi (A \,{}_h\!\oplus\, B) = (t{:=}e \fatsemi A) \,{}_{h \circ e}\!\oplus\, (t{:=}e \fatsemi B)$.

Pure nondeterministic choice $\oplus$ is similar to nondeterminism in the (non-probabilistic) sequential model. In particular, it satisfies right distributivity for sequential composition, which does not hold in previous probabilistic models, suggesting that the operator does not exploit history:

Law 3. (1) *The composition* $\oplus$ *is idempotent, commutative and associative.*
(2) *Sequential composition is associative and distributes over* $\oplus$.
(3) *The probabilistic choice* ${}_h\oplus$ *distributes over* $\oplus$.

Abort and skip are fixpoints of nondeterministic closure, which forces the current state to coincide with the original state for the assignment:

Law 4. (1) $\Box(\bot) = \bot$ (2) $\Box(\mathbb{I}) = \mathbb{I}$
(3) $\Box(t := e(s,t)) = t := e(t,t)$.

A nondeterministic closure resets the original state to be the initial state of its argument. Thus adjacent closures have the same effect as one closure. Closure distributes over convex combination with a current choosing distribution. If the second half of a sequential composition in a closure is closed, the outer closure can be decomposed into two closures in a sequential composition:

Law 5
(1) $\Box(A \fatsemi B) = \Box(\Box(A) \fatsemi B)$ (2) $\Box(A \,{}_h\oplus\, B) = \Box(\Box(A) \,{}_h\oplus\, B)$
(3) $\Box(A \oplus B) = \Box(\Box(A) \oplus B)$ (4) $\Box(A \,{}_v\oplus\, B) = \Box(A) \,{}_v\oplus\, \Box(B)$
(5) $\Box(A \fatsemi \Box(B)) = \Box(A) \fatsemi \Box(B)$.

The proof for Law 5(5) is included to illustrate the semantic reasoning style.

Proof. For all well-formed distributions d and d',

$[\Box(A \fatsemi \Box(B))](d,d')$
$\Leftrightarrow$ definitions
$\exists d_0, d_1, d_2 \cdot A(\delta, d_1) \wedge B(\delta, d_2) \wedge [\Box(B)](d_1, d_0) \wedge d_0 = d_1 \otimes d_2 \wedge d' = d \otimes d_0$
$\Leftrightarrow$ predicate calculus
$\exists d_1, d_2 \cdot A(\delta, d_1) \wedge B(\delta, d_2) \wedge d' = d \otimes (d_1 \otimes d_2)$
$\Leftrightarrow$ associativity
$\exists d_1, d_2 \cdot A(\delta, d_1) \wedge B(\delta, d_2) \wedge d' = (d \otimes d_1) \otimes d_2$
$\Leftrightarrow$ predicate calculus
$\exists d_0, d_1, d_2 \cdot A(\delta, d_1) \wedge B(\delta, d_2) \wedge$
$\quad [\Box(A)](d, d_0) \wedge d_0 = d \otimes d_1 \wedge [\Box(B)](d_0, d') \wedge d' = d_0 \otimes d_2$
$\Leftrightarrow$ definitions
$[\Box(A \fatsemi \Box(B))](d,d')$.

Thus the two programs correspond to the same relation. □

Two usual laws hold for the weakest fixpoint operator:

Law 6. (1) $F(\mu F) = \mu F$ (1) *if* $A \subseteq F(A)$ *then* $A \subseteq \mu F$.

A program is called *finite* if it consists of only abort $\bot$, closed assignment $t := e(t)$, sequential composition, closed probabilistic choice $A \,{}_v\oplus\, B$, open nondeterminism $\oplus$ and nondeterministic closure $\Box$. A finite program is equal to $\bot$, $\mathbb{I}$ or can be written as:

$$\bigoplus_i \bigoplus_{j, v_{ij}} (t := e_{ij}(t,t) \fatsemi \Box(N_{ij}))$$

where each N_{ij} is a program in normal form and for any i, we assume $\sum_j v_{ij} = 1$. Here we are using a collective form of probabilistic choices under which $A \,{}_v\!\oplus\, B$ is represented as $\bigoplus_{i,v_i} A_i$ where $i = 1, 2$, $v_1 = v$, $v_2 = 1-v$, $A_1 = A$ and $A_2 = B$. Adjacent probabilistic choices can be aggregated using Law 2(3). The following syntax represents the normal form formally.

Theorem 1. *Every finite program is semantically equal to a program in the following normal form* $\mathbf{N}$*:*

$$\begin{array}{l} \mathbf{N} ::= \mathbf{M} \mid \mathbf{N} \oplus \mathbf{N} \\ \mathbf{M} ::= \bot \mid \mathtt{II} \mid (t := e(t) \fatsemi \Box(\mathbf{N})) \mid \mathbf{M} \,{}_v\!\oplus\, \mathbf{M}. \end{array}$$

Proof. Notice that every program $M \in \mathbf{M}$ is closed according to Law 4(3) and Law 5(5).

1. Primitives are already in normal form: $\bot$, $t := e(t) \in \mathbf{N}$.
2. For two finite programs $N, N' \in \mathbf{N}$, their sequential composition $(N \fatsemi N')$ can be reduced to normal form, because by induction:
   (a) If N is a primitive, $(N \fatsemi N')$ is reducible according to Law 2(5) and Law 3(2).
   (b) Assume all $(N_i \fatsemi N')$ are reducible. Then for $M, M' \in \mathbf{M}$ such that $M = \bigoplus_{i,v_i} (t := e_i \fatsemi \Box(N_i))$ and $M' = \bigoplus_{j,v'_j} (t := e'_j \fatsemi \Box(N'_j))$, according to Law 2(4), we have $(M \fatsemi M') = \bigoplus_{i,v_i} (t := e_i \fatsemi \Box(N_i) \fatsemi M') = \bigoplus_{i,v_i} (t := e_i \fatsemi \Box(N_i \fatsemi M')) \in \mathbf{N}$. Thus in general $N = \bigoplus_i M_i$ and $N' = \bigoplus_j M'_j$, and $(N \fatsemi N') = \bigoplus_{ij} (M_i \fatsemi M'_j) \in \mathbf{N}$.
3. According to Law5(4), for all $M, M' \in \mathbf{M}$, we have $(M \,{}_v\!\oplus\, M') \in \mathbf{M}$. Thus for finite programs $N, N' \in \mathbf{N}$, their probabilistic choice is reducible: $(N \,{}_v\!\oplus\, N') = \bigoplus_{ij} (M_i \fatsemi M'_j) \in \mathbf{N}$.
4. Obviously $(N \oplus N') \in \mathbf{N}$ for all $N, N' \in \mathbf{N}$.
5. If $N \in \mathbf{N}$, then $\Box(N) = \bigoplus_{i=1}^{1} \bigoplus_{j=1,v_1}^{1} (\mathtt{II} \fatsemi \Box(N)) \in \mathbf{N}$ where $v_1 = 1$.

Thus every finite program can be transformed to normal form using just the laws, which are themselves sound in the semantic model. □

2.6 Program Verification

An assertion can be regarded as a predicate of a single distribution variable d. It is therefore more convenient to use a set of well-formed distributions to represent an assertion. If A is a program and $\mathbb{P}, \mathbb{Q} \subseteq \mathbb{D}$ are subsets of well-formed distributions, then annotation $(\mathbb{P} \;\; A \;\; \mathbb{Q}) \mathrel{\widehat{=}} \forall d, d' \cdot (d \in \mathbb{P} \wedge A(d, d') \Rightarrow d' \in \mathbb{Q})$ states that if the computation A starts from an initial distribution in $\mathbb{P}$, then the final distributions that it yields lie in $\mathbb{Q}$. As our model is relational, if all assertions $\mathbb{P} \in \mathcal{P}$ and $\mathbb{Q} \in \mathcal{Q}$ in some set clusters satisfy $(\mathbb{P} \;\; A \;\; \mathbb{Q})$, so does their universal union: $(\bigcup \mathcal{P} \;\; A \;\; \bigcup \mathcal{Q})$.

3 Two Case Studies

In this section, we consider two case studies. In the first we revisit (1) and show that the new semantic model distinguishes the two sides, just like previous models; it illustrates the style of reasoning with the model. The second case study illustrates how the properties of realistic probabilistic programs are treated. The Monty-Hall problem is introduced and modified in a manner that demonstrates the power of the model.

3.1 The Example Mentioned in Introduction

Example (1) was originally studied in [4]. The difference between the two seemingly equal programs is due, semantically, to the fact that open nondeterministic choice can make different probabilistic choices from different original states. All commands normally share the same original state, but a nondeterministic closure can alter this and reset the original state, allowing the computation within the closure to act against the recorded probabilistic choice in the original state. We consider both informal and formal versions.

Let $V = \{ x, y \}$ be the set of all program variables whose values are boolean: $C = \{ 0, 1 \}$. To distinguish the two programs, we simply need to show that from some initial distribution (*i.e.* $\langle 0, 0 \rangle$), the two programs may yield different sets of final distributions. The functional distribution $\langle 0, 0 \rangle$ denotes that the probability is 1 for $x = 0$ and $y = 0$ regardless of original state.

We first consider the program on the left. The nondeterministic closure resets the distribution with the delta distribution $\langle x, y \rangle$, equating the current state to the original state. The open nondeterministic choice performs convex combination with an arbitrary factor distribution u between the results $\langle 0, y \rangle$ and $\langle 1, y \rangle$ of the assignments $x := 0$ and $x := 1$. The result of arbitrary convex combination is convoluted with the functional initial distribution. Convolution distributes over the convex combination. The effect is equivalent to choosing between $\langle 0, 0 \rangle$ and $\langle 1, 0 \rangle$ with an arbitrary constant factor. The fair probabilistic choice combines the result distributions with equal probability. For any constant factor, the overall probability for $x = y$ is always $1/2$. Formally,

$$
\begin{array}{l}
\{ \langle 0, 0 \rangle \} \\
\Box \{ \langle x, y \rangle \} \\
\quad x := 0 \quad \{ \langle 0, y \rangle \} \quad \oplus \quad x := 1 \quad \{ \langle 1, y \rangle \} \\
\quad \{ \langle 0, y \rangle \, {}_{u}\!\oplus \, \langle 1, y \rangle \mid u \in \mathbb{U} \} \\
\{ \langle 0, 0 \rangle \otimes (\langle 0, 0 \rangle \, {}_{u}\!\oplus \, \langle 1, 0 \rangle) \mid u \in \mathbb{U} \} \\
\{ \langle 0, 0 \rangle \, {}_{c}\!\oplus \, \langle 1, 0 \rangle \mid c \in \mathbb{C} \} \\
y := 0 \; {}_{\frac{1}{2}}\!\oplus \; y := 1 \\
\left\{ (\langle 0, 0 \rangle \, {}_{c}\!\oplus \, \langle 1, 0 \rangle) \, {}_{\frac{1}{2}}\!\oplus \, (\langle 0, 1 \rangle \, {}_{c}\!\oplus \, \langle 1, 1 \rangle) \mid c \in \mathbb{C} \right\} \\
\{ d \mid d \cdot [x = y] \; = \; c/2 + (1 - c)/2 \; = \; 1/2 \}.
\end{array}
$$

The annotation for the probabilistic choice requires the universal union rule of Section 2.6 and:

$$\begin{array}{l}\{\,\langle 0,y\rangle \mathbin{\mathcal{L}\!\!\oplus} \langle 1,y\rangle\,\}\\ \quad y:=0 \quad \{\,\langle 0,0\rangle \mathbin{\mathcal{L}\!\!\oplus} \langle 1,0\rangle\,\} \quad {}_{\frac{1}{2}}\!\oplus \quad y:=1 \quad \{\,\langle 0,1\rangle \mathbin{\mathcal{L}\!\!\oplus} \langle 1,1\rangle\,\}\\ \{\,(\langle 0,0\rangle \mathbin{\mathcal{L}\!\!\oplus} \langle 1,0\rangle) \;{}_{\frac{1}{2}}\!\oplus\; (\langle 0,1\rangle \mathbin{\mathcal{L}\!\!\oplus} \langle 1,1\rangle)\,\}.\end{array}$$

On the right-hand side of Example (1), the distribution before the nondeterministic choice has equal probability for distributions $\langle 0,0\rangle$ and $\langle 0,1\rangle$. That means the arbitrary uniform factor u's values at two original states $(0,0)$ and $(0,1)$ are exploited for convex combination in the subsequent nondeterminism. So no non-zero probability for $x=y$ can be *guaranteed*. Formally,

$$\begin{array}{l}\{\,\langle 0,0\rangle\,\} \quad y:=0 \;{}_{\frac{1}{2}}\!\oplus\; y:=1 \quad \left\{\,\langle 0,0\rangle \;{}_{\frac{1}{2}}\!\oplus\; \langle 0,1\rangle\,\right\}\\ \Box\,\{\,\langle x,y\rangle\,\}\\ \quad x:=0 \quad \{\,\langle 0,y\rangle\,\} \quad \oplus \quad x:=1 \quad \{\,\langle 1,y\rangle\,\}\\ \quad \{\,\langle 0,y\rangle \;{}_{u}\!\oplus\; \langle 1,y\rangle \mid u\in\mathbb{U}\,\}\\ \left\{\,(\langle 0,0\rangle \;{}_{\frac{1}{2}}\!\oplus\; \langle 0,1\rangle)\otimes(\langle 0,y\rangle \;{}_{u}\!\oplus\; \langle 1,y\rangle) \mid u\in\mathbb{U}\,\right\}\\ \left\{\,(\langle 0,0\rangle \;{}_{c_1}\!\oplus\; \langle 1,1\rangle) \;{}_{\frac{1}{2}}\!\oplus\; (\langle 1,0\rangle \;{}_{c_2}\!\oplus\; \langle 1,1\rangle) \mid c_1,c_2\in\mathbb{C}\,\right\}\\ \left\{\,d \mid d\cdot[x=y] = c_1 \;{}_{\frac{1}{2}}\!\oplus\; (1-c_2),\ c_1,c_2\in\mathbb{C}\,\right\}.\end{array}$$

3.2 The Monty-Hall Problem

This case study illustrates how the nondeterministic demon and the computation can both benefit from the extra information recorded in the original state. The last variation of the case study is not captured by previous models.

The famous Monty-Hall puzzle (in this context, from [8]) describes a game show with a host, a contestant and three closed doors: one of them hides a car, which the contestant wishes to win, whilst the other two hide goats, which the contestant intends to avoid. The contestant begins by choosing a door, but it is not opened immediately. Instead, the host opens a door different from the one just chosen by the contestant and then offers the contestant the option of switching choices to one of the other two unopened doors. Contrary to common perception that the chance of winning the car is unchanged by whether or not the contestant switches, the correct move is to switch, and it doubles the overall chance from 1/3 (independent initial right choice among three) to 2/3 (initial wrong choice to be corrected with the host's assistance).

This puzzle and its solution involve both probability and nondeterminism. Let the variable x denote the host's initial choice (in program HC) of the door (number 1, 2 or 3) for the car, which is completely unknown to the contestant:

$$HC \;\widehat{=}\; x:=1 \sqcap (x:=2 \sqcap x:=3).$$

The variable y denotes the contestant's choice. The contestant, with no knowledge of the position of the car, chooses fairly among the three:

$$PC \;\widehat{=}\; y := 1 \;{}_{\frac{1}{3}}\!\oplus\; (y := 2 \;{}_{\frac{1}{2}}\!\oplus\; y := 3).$$

Note that this is one possible strategy for the contestant, and a program may well adopt a different (and possibly bad) strategy. The contestant's only knowledge is that the host chooses the door before the contestant's initial choice, and that the host is not prescient. The host's subsequent door opening (denoted with variable z) depends on the contestant's initial choice. If the choice is right, then the host chooses one of the two remaining goat doors; otherwise, the host chooses the only remaining goat door:

$$HC1 \;\widehat{=}\; (z := goat_1(x) \sqcap z := goat_2(x)) \;{}_{[x=y]}\!\oplus\; z := goat(x, y)$$

where the function $goat_1(x)$ returns the smaller door number other than x, $goat_2(x)$ returns the larger number, and $goat(x, y)$, defined only when $x \neq y$, returns the only other number. For example, $goat_2(2) = 3$ and $goat(1, 2) = 3$. The contestant's second choice results either in stay $ST \;\widehat{=}\; \mathbb{I}$ or switch $SW \;\widehat{=}\; y := goat(y, z)$. The game corresponds to the computation:

$$HC \fatsemi PC \fatsemi HC1 \fatsemi ?\,.$$

A routine calculation guarantees probability 1/3 if the question mark is replaced by ST but 2/3 if that is replaced by SW. Note that the position of the car must not depend on the contestant's initial choice. The host's placement of the car *after* the contestant's choice is:

$$PC \fatsemi HC \fatsemi HC1 \fatsemi ?\,.$$

It is *unknown* whether the (nondeterministic) host places the car against or for the contestant's interests, or indeed chooses neutrally. As a result, no matter what the contestant does in the end, the strategy cannot guarantee even a small probability of success. That phenomenon is modelled correctly by both this and previous models.

Now consider a less honest host who, though setting the car before the contestant, has detected the contestant's tendency to switch. He secretely tries to move the car to the door of the contestant's first choice with probability 1/3 (no move if the contestant is right):

$$HC2 \;\widehat{=}\; (x := y \;{}_{\frac{1}{3}}\!\oplus\; \mathbb{I}).$$

The host performs this dishonest act after opening a door but before the contestant's final choice:

$$HC \fatsemi PC \fatsemi HC1 \fatsemi HC2 \fatsemi ?\,.$$

Now the probability of initial correctness and staying with the original choice is increased to $1/3 + (2/3 \times 1/3) = 5/9$, while the probability of success for switching drops to 4/9.

However, an alert contestant decides to stay whenever he detects any noise of car movement (*i.e.* detecting a change of state; we assume that the host is not devious enough to move the car around behind the same door) but to switch otherwise:

$$AC \;\widehat{=}\; SW \;{}_{\delta}\!\oplus\; ST.$$

The contestant performs that scrutiny by comparing the *original state*, immediately after the host opens a door, with the *current state* before the final decision:

$$HC \fatsemi PC \fatsemi HC1 \fatsemi \Box(HC2 \fatsemi AC)\,.$$

The car moves with probability $2/3 \times 1/3 = 2/9$ when the contestant's initial choice was wrong. It is always favourable for the contestant to stay after detecting noise. Switching yields probability $2/3 \times 2/3 = 4/9$ of success for the contestant's unswitched initial wrong choice (greater than the success probability $1/3$ for staying). The overall probability of success of AC is $2/9 + 4/9 = 2/3$. Formally:

$$
\begin{array}{ll}
\{\,\langle 1,1,1\rangle\,\} & \\
\Box\Box\,\{\,\langle x,y,z\rangle\,\} & \text{Host's choice} \\
\quad x := 1 \oplus\; (x := 2 \oplus\; x := 3) \fatsemi & \\
\{\,\langle 1,1,1\rangle\;{}_{c_1}\!\oplus\; (\langle 2,1,1\rangle\;{}_{c_2}\!\oplus\; \langle 3,1,1\rangle) \mid c_1, c_2, \in \mathbb{C}\,\} & \\
y := 1\;{}_{\frac{1}{3}}\!\oplus\; (y := 2\;{}_{\frac{1}{2}}\!\oplus\; y := 3) \fatsemi & \text{Player's choice} \\
\{\,d \mid d\cdot[x=y] = 1/3\,\} & \\
(z := goat_1(x) \sqcap z := goat_2(x))\;{}_{[x=y]}\!\oplus\; z := goat(x,y) \fatsemi & \text{Host opens a door} \\
\{\,d \mid d\cdot[x=y] = 1/3 \wedge d\cdot[y \neq x = goat(y,z)] = 2/3\,\} & \\
\Box\,\{\,\langle x,y,z\rangle\,\} & \\
\quad x := y\;{}_{\frac{1}{3}}\!\oplus\; \text{I\!I} \fatsemi & \text{Car move} \\
\quad \left\{\,\langle y,y,z\rangle\;{}_{\frac{1}{3}}\!\oplus\; \langle x,y,z\rangle\,\right\} & \\
\quad y := goat(y,z)\;{}_{\delta}\!\oplus\; \text{I\!I} & \text{Alert contestant's choice} \\
\{\,d \mid d\cdot[x=y] = \frac{2}{3}\times\frac{1}{3} + \frac{2}{3}\times\frac{2}{3} = 2/3\,\}. &
\end{array}
$$

4 Related Work

This work is most closely related to the distributional model of He *et al.* [4] which appeared more than two decades after Rabin's demonstration [11] of the remarkable effectiveness of probabilistic algorithms. The obvious difference is that in our model distributions are conditional (depending on two states rather than just one). That has an interesting consequence for the definition of sequential composition. In [4] the definition is complicated by having to take the average over each intermediate distribution in the composition; our definition is simply composition of binary relations as a result of our use of conditional distributions. That makes our definition more properly 'relational' in the style of UTP [5]. Rabin's paper was followed more closely by a succession of interesting

probabilistic algorithms and of logics to facilitate model checking of probabilistic properties. For example Hansson and Jonsson [3] incorporate both time and probability. However in most of that work (demonic) nondeterminism is *replaced* by probabilism.

Our interest is in semantic models and (sound) laws for the top-down incremental derivation of an implementation from its specification. That requires nondeterminism (seen as arising from specification and modelling) *as well as* probabilism. McIver and Morgan's textbook [8] provides the fundamentals and indicates how a theory in the Dijkstra-Hoare style unfolds. It also contains Galois connections relating He *et al.*'s distributional model to the relational and predicate-transformer models of nonprobabilistic sequential programs and to the expectation-transformer model of probabilistic (nondeterministic) programs.

In order to handle original states we have adopted the view standard in mathematics but less so in computation where it has great relevance—that a conditional probability $P(A|B)$ may be used to update knowledge about $P(A)$ (for example by Bayes's formula) in the light of further, in our case sequentially provided, information. The idea of using conditional probabilities for a semantics of probabilistic programs, is not new. Incisive use of it has been made, notably, by Panangaden [10] and Ying [17]. Pananganden makes a convincing case that conditional probability distributions are the counterpart—in general—of 'probabilistic relations'. His treatment is aimed at the more general continuous case, but his insights apply here. It would be interesting to calculate his duality starting from our (nonstandard) state-based model to see what transformer model results; also to apply, to the model proposed here, the ideas captured in those extensions to 'probabilistic' predicates or relations, both at the level of the types of our semantics and in the monadic setting. Ying, on the other hand, uses conditional probability as the basis for a semantics of guarded-command-like programs with angelic choice and (of course) demonic choice but without recursion, iteration or explicit probabilistic choice that has the strength to support a refinement calculus. His models extend the distributional and expectation transformer models, by considering instead probabilistic predicates. As a result his semantics makes finer distinctions between programs and he is able to introduce a refinement relation that is probabilistic rather than Boolean in nature.

Varacca and Winskel [16] give an elegant analysis of how the monads for probability and nondeterminism might be combined. They contrast the distributive combination of the two (used, for instance, by Mislove *et al.* [7]) with their combination after modifying the probabilistic monad to contain only affine identities (hence ensuring the result can be lifted to the power set).

As indicated by our treatment of the Monty Hall problem and its variations, a framework incorporating probability and nondeterminism addresses issues of secrecy and information flow. Most closely related to our approach in that direction is the work of Morgan [9]. Whilst our nondeterminism has limited memory, his has limited vision, since program state is separated into visible and hidden parts; but then his novel refinement relation becomes the primary tool (whilst we retain the standard equivalence connecting nondeterminism and refinement:

$P \sqsubseteq Q$ iff $P = P \sqcap Q$). Morgan's approach is demonstrated on Chaum's 'Dining Cryptographers' and Rivest's 'Oblivious Transfer'. Pananganden's work mentioned above also provides an approach to the analysis of information flow in security protocols; see for instance [1] which contains further references.

There is a very much greater literature devoted to the difficult topic of probability in reactive and parallel programs. See [14] for a survey to 2004, in the setting of probabilistic automata. Inevitably some of those contributions are relevant to the sequential case. Typically there demonic nondeterminism is viewed as freedom to be exploited by a scheduler [13]. Whilst true as far is that is able to be exploited in the sequential case (for 'scheduler' read 'implementer'), here it is not the overriding consideration, which is the interaction between probabilistic choice and sequential composition. Focusing on that, the sequential case might be viewed as a convenient starting point for a later study of reactive nondeterministic and probabilistic computation, in which the interaction between nondeterminism and probability is studied without the concerns of deadlock, divergence and so on. Contributions to the reactive case that will no doubt be influential due to the combination of nondeterminism and probability include the work of Mislove *et al.* [7] and Tix *et al.* [15] which construct models of process algebra with nondeterminism and probability, as a solution to a domain equation using the Plotkin powerdomain. Also the probabilistic automata of Segala [13] embody important principles. As theories for reactive probabilistic systems account largely for the various forms of 'process testing', they appear currently to be surprisingly divergent from those for sequential probabilistic systems.

5 Conclusion and Further Work

This paper has introduced a relational probabilistic model containing both probabilistic choice and nondeterministic choice. The standard demonic nondeterministic choice is decomposed into two operators: one that performs convex closure and the other that performs nondeterministic closure. The introduction of original states and consequent use of conditional probability distributions help to relate sequential specifications with some past state, whilst at the same time ensuring that sequential composition remains relational.

That clarifies what nondeterminism really does, and facilitates further generalisations. One such is to allow commands, like assignment and probabilistic choice, to observe original state. That allows a later computation to perform compensation back to the starting point of the closest nondeterministic closure. Another possibility is to strengthen the manner in which a nondeterministic choice can exploit further history by introducing more original states; then an open nondeterministic choice can act against the probabilistic choices at several points set by nested nondeterministic closures. Such generalisations become possible only after we explicitly reveal what nondeterministic choices really do and they have, as we have seen, important application beyond the realm of probabilistic programs (for example in the design of security protocols, where the adversary can be regarded as nondeterministic if it is unknown whether it can observe certain information and take advantage of the observation).

Sequential composition of programs has been modelled as composition of binary relations, but at the expense of a mild complexity in the semantics. In the distributional model, the reverse is the case: the definition of sequential composition requires imposition of healthiness conditions. As a result, our version may prove easier in the so-far-unachieved goal of unifying probabilism with other programming constructs in the style of *Unifying Theories of Programming*, [5].

References

1. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: On the Bayes risk in information-hiding protocols. In: Proceedings of the 20th IEEE Computer Security Foundations, pp. 341–354. IEEE Computer Society, Los Alamitos (2007)
2. Fernandez, L., Piron, R.: Should She Switch? A Game Theoretic Analysis of the Monty Hall Problem. Mathematics Magizine 72(3), 214–217 (1999)
3. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
4. He, J., Seidel, K., McIver, A.K.: Probabilistic models for the guarded command language. Science of Computer Programming 28(2), 171–192 (1997)
5. Hoare, C.A.R., He, J.: Unifying Theories of Programming. Prentice Hall, Englewood Cliffs (1998)
6. McIver, A.K., Morgan, C.C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science. Springer, Heidelberg (2005)
7. Mislove, M., Ouaknine, J., Worrell, J.: Axioms for probability and nondeterminism. ENTCS 96, 7–28 (2004)
8. Morgan, C.C., McIver, A.K., Seidel, K.: Probabilistic Predicate Transformers. ACM TOPLAS 18(3), 325–353 (1996)
9. Morgan, C.C.: The shadow knows: Refinement of ignorance in sequential programs. In: Uustalu, T. (ed.) MPC 2006. LNCS, vol. 4014, pp. 359–378. Springer, Heidelberg (2006)
10. Panangaden, P.: Probabilistic relations. In: Baier, C., Huth, M., Kwiatkowska, M.Z., Ryan, M. (eds.) PROBMIV 1998, pp. 59–74 (1998)
11. Rabin, M.O.: Probabilistic algorithms. In: Traub, J.F. (ed.) Algorithms and Complexity: New Directions and Recent Results, pp. 21–39. Academic Press, London (1976)
12. Robinson, D., Morgan, C.C.: Restricted Demonic Choice for Modular Probabilistic Programs. In: ESSLI PLRC Workshop 1998 (1998)
13. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. In: Jonsson, B., Parrow, J. (eds.) CONCUR 1994. LNCS, vol. 836, pp. 1–43. Springer, Heidelberg (1994)
14. Sokolova, A., de Vink, E.: Probabilistic automata: system types, parallel composition and comparison. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 1–43. Springer, Heidelberg (2004)
15. Tix, R., Keimel, K., Plotkin, G.: Semantic domains for combining probability and non-determinism. ENTCS 222, 3–99 (2009)
16. Varacca, D., Winskel, G.: Distributing probability over non-determinism. Mathematical Structures in Computer Science 16(1), 87–113 (2006)
17. Ying, M.: Reasoning about probabilistic sequential programs in a probabilistic logic. Acta Informatica 39, 315–389 (2003)