Cost analysis of games, using program logic*

Carroll Morgan† and Annabelle McIver‡

Abstract

Recent work in programming semantics has provided a relatively simple probablistic extension to *predicate transformers*, making it possible to treat small imperative probabilistic programs containing both demonic and angelic nondeterminism [11, 12, 20]. That work in turn was extended to provide a probabilistic basis for the *modal μ-calculus* [13], and leads to a *quantitative* μ-calculus [16, 18].

Standard (non-probabilistic) μ-calculus can be interpreted either 'normally', over its semantic domain, or as a two-player game between an 'angel' and a 'demon' representing the two forms of choice. It has been argued [23] that the two interpretations correspond.

Quantitative μ-calculus can be interpreted both ways as well, with the novel interpretation being the second one: a *probabilistic* game involving an angel and a demon. Each player seeks a strategy to maximise (resp. minimise) the game's 'outcome', with the steps in the game now being stochastic. That suggests a connection with Markov decision processes, in which players compete for high (resp. low) 'rewards' over a Markov transition system.

In this paper we explore 'the Markov connection', showing for example how discounted Markov decision processes (MDP's) and terminating MDP's can be written as quantitative μ-formulae. The 'normal' interpretation of those formulae (i.e. over the semantic domain) then

*Presented at the 8th Asia-Pacific Software Engineering Conference (APSEC 2001), 4–7 December 2001, University of Macau, Macau SAR, China.

†Dept. of Computer Science and Engineering, University of New South Wales, 2052 Australia: *carrollm@cse.unsw.edu.au*

‡Department of Computer Science and Mathematics, Macquarie University, 2109 Australia: *anabel@mpce.mq.esu.au*

gives a more direct access to existence theorems than the presentation usually associated with MDP's.

Our technical contribution is to explain the coding of MDP's as quantitative μ formulae, to discuss the extension of the latter to incorporate 'rewards', and to illustrate the resulting reformulation of existence theorems. In doing so we use an existing (but unpublished) extension to our normal probabilistic semantics, the 'Lamington Model'; it is described in the appendix.

1 Introduction

The behaviour of programs or systems depends on the conditions under which they operate. In many cases those conditions will be imprecisely specified and, for the analysis of such systems to be useful at all, it is vital that as much as possible of the available information is taken into account.

Whilst some uncertainty can never be factored out — examples include cases where arbitrary behaviour can be traced to unpredictable system-level decisions — there are many circumstances in which an environment's behaviour can be partially (and usefully) described using probability. Examples include (hybrid) systems having hardware components with known failure rates, or protocols providing resource-access subject to users' fluctuating demands which, in turn, are perhaps related (although only statistically so) to the time of day.

Thus we can distinguish between 'quantifiable' and 'unquantifiable' uncertainty. In recent years much research effort has focussed on how to fuse the two notions, so that valuable indications of behaviour can be deduced taking both into account. *Unquantifiable* uncertainty is familiar in theories of computing and is modelled by 'demonic nondeterminism', a term employed to mean that a system cannot be relied upon to proceed in any single and repeatable 'determined' way: the best that can be done in these situations is to include all possibilities, both the 'bad' and the 'good'. *Quantifiable* uncertainty is less well known in the computing literature; but when it appears it is usually dealt with by incorporating standard probability theory into existing mathematical models of programs.

Early mathematical treatments containing both probability and nondeterminism[1] sought to introduce *probabilistic models* of distributed systems,

[1]When we write "nondeterminism" on its own, we mean demonic rather than proba-

which went a long way towards clarifying the many issues related to how probability and nondeterminism should interact. A major contribution of the work was to provide specification languages and verification techniques for probabilistic properties in general [22] and temporal properties in particular [24, 1, 7]. Examples of such properties include "a given set of system states will be reached with probability 1" and "the probability that a given property is invariant". Later work extended the application areas to include average expected times between events and shortest-path problems [3].

A key component of those treatments was to provide actual algorithms for computing the properties specified, and many model-checking systems are based on the ideas put forward in that early work. However reasoning (even about small systems) within the given framework, being primarily model-based, can be very intricate indeed.

There is a dual approach to this specification and analysis problem: its motivation is to expose the properties of a probabilistic system using a *quantitative logic* of probabilistic programs, in which the expressions include real values that reflect the probabilistic transitions in the underlying system. As with program logic generally, it leads to simpler specification and analysis techniques.

Nondeterminism proves to be a significant hurdle in the logical approach, however — Harel's [5] and Kozen's [12] logics for example do not include it, and though Kwiatkowska's [9] and Bianco's [2] logics have an interpretation over models containing both probability and nondeterminism, they do not investigate the foundations of the axiom system. As far as we are aware, the *probabilistic weakest preconditions*, extended by Morgan *et al.* [20] from Kozen's original approach [11], was the first logic to characterise a model containing both probability and demonic nondeterminism. From it, many powerful analysis techniques have been derived.

From the mathematical standpoint, the models in all of the above share many of the features found in *Markov Decision Processes* (*MDP*'s). That comes as no surprise, since *MDP*'s were originally formulated to address realistic problems related to resource contention within a probabilistic context. A (*non-competitive*) *MDP*, in its most abstract form, describes the actions of a solitary gambler and the consequences of his selected strategy in placing bets. Probabilistic programs can similarly be seen as such games: the nondeterminism in the program represents a range of strategies a gambler

bilistic.

might follow, whilst the probabilistic transitions correspond to the games' underlying conditions of chance.

The use of *MDP*'s brings with it a long-established body of algorithmic techniques, based on linear programming, for quantifying certain well-defined properties; again however, even simple arguments can be intricate and difficult to follow — largely because of the need to include all the details of each *MDP*. There is no built-in facility of abstraction.

Abstraction, however, is the essence of logic. In this paper we review some properties of *MDP*'s from a logical perspective, and we find that there are many advantages in doing so. First, we show that for the purposes of assessing game strategies, it indeed suffices to encapculate the complexities of the *MDP*'s' infrastructure by concentrating on their *logical properties*. In doing so we find an immediate benefit in the presentation, because the focus shifts towards the properties required for games to be properly defined. The more substantial benefit, however, is that the logic has a natural interpretation within a domain of probabilistic programs, and that allows us to give a domain-theoretic 'spin' to the proofs — one effect of which is to simplify the proofs of key existence theorems.

In Sections 2 and 3 we review some basic theory of *MDP*'s and probabilistic programs. In Sec. 4 we concentrate on their similarities; and finally in Sec. 5 we exploit the similarities to apply domain-theoretic techniques from program logic to the proof of a basic theorem of *MDP*'s, the existence of optimal memoryless strategies. Sec. 6 concludes.

We use left-associating dot '.' for functional application, and write quantifications in the order *bound variable* then *constraint* then *term*, with scoping always explicit, given either by parentheses $(\cdots)$ or set-comprehension braces $\{\cdots\}$: thus the set of positive cubes is $\{i{:}\,\mathbb{Z} \mid i > 0 \cdot i^3\}$.

2 Non-competitive *MDP*'s: the basics

2.1 Markov Decision Processes

In this section we set out the basic theory of *MDP*'s, following closely the approach taken by Filar and Vrieze [6]. Let S be a *finite state space*, L a finite set of *labels*, and let $\overline{S}$ be the set of *discrete probability distributions*

over S.[2]

Definition 2.1 A finite-state *Markov Decision Process* over S is a pair (ρ, ω), where ρ is a transition system of type $S \times L \to \overline{S}$ and ω is a reward function of type $S \times L \to \mathbb{R}$. The transition system ρ maps every (initial) state s in S and label l in L to a probability distribution $\rho.s.l$ over final states, whilst ω specifies an immediate reward $\omega.s.l$ following the selection of l in initial state s. □

The game metaphor, which complements the above definition, provides a useful intuition based on a gambler's playing a game of chance:

- From his initial position s he may choose from a selection of S-labelled dice; each die is distinguished by a label l and is biased according to the distribution $\rho.s.l$.
- Each choice incurs an immediate cost or reward of $\omega.s.l$.
- His total expected winnings are determined by his immediate reward and the final state s' he reaches after rolling his chosen die: if there is a payoff function $\phi\colon S \to \mathbb{R}$, then he wins $\phi.s'$ plus the reward.

Thus if the gambler plays from initial state s, and chooses to roll the die labelled l, then his total *expected* payoff is given by

$$\omega.s.l + \int_{\rho.s.l} \phi \quad ,$$

where in general we write $\int_d \phi$ for the expectation of random variable ϕ over distribution d.[3] It is simply the immediate payoff added to his expected reward due to the payoff function ϕ.

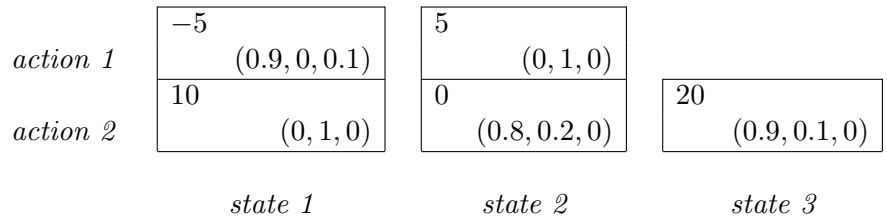
Typically, in the context of *MDP*-cost problems, the gambler plays either to win, or to lose:[4]

[2] Later we will extend $\overline{S}$ to include *sub*-distributions, those that sum to no more than 1 (rather than exactly to 1).

[3] In the finite case, as here, it is simply

$$\sum_{s':S} (\rho.s.l.s' \times \phi.s') \ .$$

[4] At this point the metaphor wears a bit thin: gamblers who don't *want* to win are few and far between. Nevertheless the idea is still useful in computing applications, where the role of the gambler is taken by a demon who acts as a crook trying to lose someone else's money.



In this representation a box portrays an action in a state and its reward/transition (upper-left/lower-right) consequences. For example, in the above the choice of action 1 in state 1 results in a cost (negative reward) of $\omega.1.1 = -5$ and transition probabilities $\rho.1.1.1 = 0.9$, $\rho.1.1.2 = 0$ and $\rho.1.1.3 = 0.1$.

Figure 1: A summable Markov Decision Process [6, Example 2.1.1 p10]

Definition 2.2 Given the *MDP* (ρ, ω) and payoff function ϕ, a gambler's *least expected reward* from initial state s is given by the function $\underline{val}$:

$$\underline{val}.\rho.\omega.\phi.s \quad := \quad (\sqcap\, l{:}\, L \cdot \omega.s.l + \int_{\rho.s.l} \phi \quad) \ .$$

□

Similarly, a gambler's *greatest expected reward* from initial state s is given by the function $\overline{val}$:

Definition 2.3

$$\overline{val}.\rho.\omega.\phi.s \quad := \quad (\sqcup\, l{:}\, L \cdot \omega.s.l + \int_{\rho.s.l} \phi \quad) \ .$$

□

We illustrate the above definitions using an example due to Filar, reproduced here in Fig. 1.

Assume the payoff function is given by

$$\phi.s_1 := 10, \ \phi.s_2 := -20, \ \phi.s_3 := 30 \ .$$

The gambler's least expected winnings from (initial) s_1 are:

$$
\begin{array}{ll}
 & \underline{val}.\rho.\omega.\phi.s_1 \\
= & (-5 + 0.9{\times}10 + 0.1{\times}30) \ \sqcap \ (10 + 1{\times}(-20)) \\
= & 7 \sqcap -10 \\
= & -10 \ .
\end{array}
$$

And a gambler's greatest expected winnings from (initial) s_2 are:

$$
\begin{array}{ll}
 & \overline{val}.\rho.\omega.\phi.s_2 \\
= & (5 + 1{\times}(-20)) \ \sqcup \ (0 + 0.8{\times}10 + (0.2{\times}(-20)) \\
= & -15 \sqcup 4 \\
= & 4 \ .
\end{array}
$$

2.2 Accumulation and discounted-cost games

Suppose now that the gambler plays repeatedly, accumulating his reward as he goes. At each time t (discrete, say in $\mathbb{N}$) he chooses a label determining his next move; let his *strategy* be $\sigma\colon \mathbb{N} \times S \to L$ representing 'in advance' the choice of label he would make from s if he were there at time t. The gambler's problem now is to evaluate the effectiveness of his strategy, were he to follow it, in terms of his expected payoffs. A strategy σ, transition system ρ and an initial state s together determine a well-defined probability distribution over sequences of states in S that will be observed by the gambler as he plays the game. From that distribution the expected accumulated payoff can be computed, although in general it might be undefined in some cases. Here, however, the only games that we consider are those where the expected accumulated payoff does converge to a limit.

One such game that guarantees convergence is the *discounted-cost* game, which we describe as follows.

Definition 2.4 A *discounted MDP* is a tuple (ρ, ω, β) where (ρ, ω) is an *MDP* and the extra component $\beta\colon \mathbb{R}$ with $0 \leq \beta < 1$ is the *discount factor*. Play is according to a strategy σ as before, but the rewards are 'discounted' by the time at which they occur: a reward at time t is worth only β^t of its nominal value as given by ω. □

The discount has the effect that the later a reward is received the less it is worth: its value declines at a fixed percentage. Suppose the gambler starts

playing in state s^0 at time 0: the discount is ineffective for his first move (it is β^0), so that his immediate reward is $\omega.s^0.(\sigma.0.s^0)$ as before.

For his second move, from some state s^1 chosen probabilistically from the distribution $\rho.s^0.(\sigma.0.s^0)$, he will receive (only) $\beta \times \omega.s^1.(\sigma.1.s^1)$; therefore his *expected* reward at step 1 is[5]

$$\beta \times \int_{\rho.s^0.(\sigma.0.s^0)} \omega.s^1.(\sigma.1.s^1)\ \mathrm{d}s^1\ .$$

Similar — but increasingly complex — expressions give the expected reward for later steps.

In general we write $val\ {}_t^{\sigma,\beta}$ for the expected (discounted) winnings at time t, with strategy σ and discount factor β. (In the presence of a strategy, the question of maximum or minimum does not arise.) The gambler's overall expected winnings, were he to play according to strategy σ, is thus the sum of the individual winnings. For that we omit the t subscript:

$$val\ {}^{\sigma,\beta} \quad := \quad \sum_{0 \le t} val\ {}_t^{\sigma,\beta}\ .$$

By abstracting from the strategy, we reintroduce the issue of playing to win, or to lose:

Definition 2.5 Given the discounted *MDP* set out in Def. 2.4, a gambler's *least expected reward* is

$$\underline{val}\ {}^{\beta} \quad := \quad (\sqcap \sigma \cdot val\ {}^{\sigma,\beta})\ .$$

□

Definition 2.6 A gambler's *greatest expected reward* is

$$\overline{val}\ {}^{\beta} \quad := \quad (\sqcup \sigma \cdot val\ {}^{\sigma,\beta})\ .$$

□

The above 'ambitious' quantifications, over all possible strategies, suggest however that difficulties may await anyone wishing to compute the maximum/minimum reward in specific cases. It turns out that those difficulties can be overcome relatively easily in the special case that a gambler can achieve his optimal reward by following a *memoryless strategy*.

[5] In the usual way, we read "$\mathrm{d}s^1$" as functional abstraction of the expression $\omega.s^1.(\sigma.1.s^1)$ over its parameter s^1.

Definition 2.7 A strategy σ is said to be *memoryless* if it is time-independent, *i.e.* if σ can be written as a function of s alone. □

The significance of memoryless strategies is that time can be factored out of the calculation, to a large extent, thus placing the problem back in the context of finite state spaces. And it means that gamblers who can achieve their optimal reward by playing such strategies can evaluate their optimal expected payoffs using linear-programming techniques.

It turns out that for discounted-cost games, memoryless strategies are always applicable, and a crucial step is to prove it. The next theorem sketches the details of the traditional proof.

Theorem 2.8 A gambler playing the discounted game set out in Def. 2.4 can achieve his least (greatest) expected reward by following a memoryless strategy.
Proof: The standard proof uses linear programming techniques, and proceeds in two stages. For full details refer to Filar [6].

1. In the class of memoryless strategies, there one that is optimal (Corollary 2.3.2, page 28).

2. The solution (1) is in fact optimal in the class of *all* strategies (Theorem 2.7.2, page 58).

Note that Filar deals with maximum rewards; but minimal rewards can be treated similarly. □

In the remainder of this paper we re-evaluate the above basic material from a programming point of view. We illustrate the new methods by giving a shorter proof, in Sec. 6, of Thm. 2.8 above.

3 Probabilistic programs: the basics

Probabilistic programs can be both nondeterministic and probabilistic, so that a model must account for both kinds of behaviour. In its simplest form [8, 20] a probabilistic program can be thought of as an unlabelled transition system.

Definition 3.1 The space $(\mathcal{R}S, \sqsubseteq)$ of *probabilistic programs* is an ordered set of functions from (initial) states S to sets of distributions over (final) states $\overline{S}$; thus $\mathcal{R}S$ is $S \to \mathbb{P}\overline{S}$. The refinement order $\sqsubseteq$ between programs is defined

$$r \sqsubseteq r' \quad \text{iff} \quad (\forall s\colon S \bullet r.s \supseteq r'.s) \ .$$

□

The operational interpretation of a probabilistic program is as a state-updating mechanism that operates in two distinct stages. In the first stage a "demon" selects a probability distribution from the set given by $r.s$; and in the second a probabilistic choice of final state is made according to that distribution. Thus the demon's choice represents demonic nondeterminism — arbitrary, unpredictable behaviour. Program refinement serves to reduce the demon's choice — if $r \sqsubseteq r'$ then the range of demonic choices available in r' is only a subset of those available in r.

There is a semantics dual to $\mathcal{R}S$ which generalises the Hoare/Dijkstra program logic well-known in formal program development. Recall the 'weakest pre-condition semantics' for programs: for program *prog* and pre- and post conditions respectively *pre* and *post*,

$$pre \quad \Rightarrow \quad wp.prog.post$$

holds iff for any intial state satisfying *pre*, the program *prog* is guaranteed to terminate in a state satisfying *post*. That *wp*-semantics is equivalent to (actually, generalises) the relational model of programs, but is more effective in practice for analysing correctness of actual programs.

We extend *wp*-semantics to probabilistic programs by replacing 'is guaranteed to terminate' by 'terminates with probability at least p'. To make this generalisation effective, it turns out that we need to cast the observations in terms of random variables, or *expectations*.[6]

Let $\mathcal{E}S$ — for *expectations over* S — be the set of real-valued functions over S.

Definition 3.2 Given a program r in $\mathcal{R}S$ and ϕ in $\mathcal{E}S$, we define the *probabilistic least pre-expectation* $plp.r.\phi$ in $\mathcal{E}S$ as follows:

$$plp.r.\phi.s \quad := \quad (\sqcap\, d\colon r.s \bullet \int_d \phi) \ .$$

[6]In fact the use of expectations is precisely the notion that allows us to have a simple joint logic for nondeterminism and probability at all. Interested readers are referred to [20] for details.

□

The real-valued quantities associated with Def. 3.2 are interpreted as the *least* expected values with respect to a program r and a 'payoff' function ϕ.

In this scheme predicates are embedded by their *characteristic functions*: given a predicate b over S, the characteristic function $\overline{b}$ is defined to be 1 exactly at those states satisfying b, and 0 otherwise. With that convention probabilistic observations may be recovered due to the fact (from standard probability theory) that the expected value of the characteristic function is the same as the probability assigned to the corresponding predicate. For example, if *prog* is a probabilistic program and $p \;\leq\; plp.prog.\overline{b}.s$, then we know that the final state satisfies b with probability at least p whenever *prog* executes from s.

The maximum expected value can be defined similarly.

Definition 3.3 Given a program r in $\mathcal{R}S$ and ϕ in $\mathcal{E}S$, we define the *probabilistic greatest pre-expectation* $pgp.r.\phi$ in $\mathcal{E}S$ as follows:

$$pgp.r.\phi.s \quad := \quad (\sqcup\, d\colon r.s \cdot \int_d \phi) \ .$$

□

In fact *plp* and *pgp* determine each other via the equivalence

$$-plp.r.(-e) \quad = \quad pgp.r.e \ , \tag{1}$$

provided we allow negative expectations.

Given a program r, both $plp.r$ and $pgp.r$ are examples of functions which transform expectations. We call such funtions *expectation transformers* and, as a generalisation of Dijkstra's predicate transformers [4], they are similarly effective for specifying and proving properties about probabilistic programs.

Definition 3.4 The space $(\mathcal{T}S, \sqsubseteq)$ of *expectation transformers* is the set of monotone functions $\mathcal{T}S$ defined $\mathcal{E}S \to \mathcal{E}S$ and ordered as follows:

$$t \sqsubseteq t' \quad \textit{iff} \quad (\forall \phi\colon \mathcal{E}S \cdot t.\phi \Rrightarrow t'.\phi) \ ,$$

where we write $\Rrightarrow$ between expectations for "is everywhere no more than". As such, it generalises implication $\Rightarrow$ between characteristic functions; and relations $\Lleftarrow$ and $\equiv$ between expectations are defined similarly. □

For all transformers t in $\mathcal{T}S$, expectations ϕ, ϕ' in $\mathcal{E}S$ and reals $c \geq 0$ we have

1. Monotonicity: if $\phi \Rrightarrow \phi'$ then $t.\phi \Rrightarrow t.\phi'$.
2. Feasibility: $t.\phi \quad \Rrightarrow \quad \sqcup \phi$.
3. Scaling: $c \times t.\phi \quad \equiv \quad t.(c \times t.\phi)$.
4. ${}_p\oplus$-subdistribution: $t.\phi \ {}_p\oplus \ t.\phi' \quad \Rrightarrow \quad t.(\phi \,{}_p\oplus \phi')$.
5. $\ominus$-subdistribution: $t.\phi \ominus \underline{c} \quad \Rrightarrow \quad t.(\phi \ominus \underline{c})$.

Figure 2: The *sublinearity* conditions [20, Fig.4 p.342]. (See also Def. A.1.)

But only a subset of the expectation transformers correspond to images of $\mathcal{R}S$ in *plp*; the properties characterising those images are known collectively as *sublinearity*, and are shown in Fig. 2.[7] The following theorem formally sets out the position.

Theorem 3.5 Any transformer t in $\mathcal{T}S$ is sublinear if and only if it is the image of some program r in $\mathcal{R}S$.
Proof: See Morgan *et al.* [20] □

Thm. 3.5 forms the basis for a logic of probabilistic programs, in that facts about programs can be proved using these axioms.

To illustrate the logic, we consider the following program, which is based on the *MDP* example from Fig. 1:

Let *prog* be the program **if**

$$\begin{array}{lcl} s_1 & \to & (s_1 \ {}_{0.9}\oplus \ s_3) \ \sqcap \ s_2 \\ s_2 & \to & s_2 \ \sqcap \ (s_1 \ {}_{0.8}\oplus \ s_3) \\ s_3 & \to & s_1 \ {}_{0.9}\oplus \ s_2 \end{array}$$

fi ,

where for brevity we omit $s =$ and $s{:=}$ in the program text.

Given post-expectation ϕ defined by

$$\phi.s_1 = 2, \ \phi.s_2 = -3, \ \phi.s_3 = 1 \ ,$$

we can calculate, for initial state s_1:

[7] The corresponding notion for standard programs is that only *conjunctive* transformers correspond to relations.

$$
\begin{array}{ll}
 & \mathit{plp.prog'.\phi.s_1} \\
= & (0.9 \times 2 + 0.1 \times 1) \sqcap -3 \\
= & -3 \ .
\end{array}
$$

Similar calculations reveal the other values of the pre-expectation:

$$\mathit{plp.prog'.\phi.s_2} = -3, \qquad \mathit{plp.prog'.\phi.s_3} = 1.5 \ ,$$

giving overall that the expected reward is -3 if *prog′* executes from either state s_1 or s_2 and is 1.5 if it executes from s_3.

We can now investigate the similarities between probabilistic programs and *MDP*'s.

4 Programs and *MDP*'s

From the last section, it is clear that *MDP*'s' transition systems have much in common with programs in $\mathcal{RS}$: in fact the main differences lie in the labelling of the transitions in the *MDP*'s and in the inclusion of the refinement order in $\mathcal{RS}$. We show firstly that in many situations the labelling is an unnecessary notational encumbrance; and secondly that exposing the fundamental order existing between transition systems makes available some nice proof techniques present in elementary domain theory, but absent in traditional presentations of *MDP*'s. We begin by investigating how far *MDP*'s and the program model $\mathcal{RS}$ are similar.

To start, we define a function $_{mToR}$, which takes a relation ρ to a program by simply forgetting the labels.

Definition 4.1 Let (ρ, ω) be an *MDP*. We define $_{mToR}.\rho$ in $\mathcal{RS}$ as follows:[8]

$$_{mToR}.\rho.s \quad := \quad \{l\colon L \bullet \rho.s.l\} \ .$$

□

Note that $_{mToR}$ 'forgets' the reward; much of our work to come will be in order to reinstate it.

It is easy to see that the *MDP*'s labels are important only if the payoff function varies between transitions. We say that an *MDP* (ρ, ω) is ω-*constant* if its payoff function doesn't depend on the label chosen.

[8]Strictly speaking we should impose closure conditions on the result space of $_{mToR}$, but for convenience here we omit them. Their details are available elsewhere [20].

Lemma 4.2 If (ρ, ω) is an ω-constant *MDP* then

$$\underline{val}.\rho.\omega.\phi \quad = \quad \omega + plp.(_{mToR}.\rho).\phi \ ,$$

and

$$\overline{val}.\rho.\omega.\phi \quad = \quad \omega + pgp.(_{mToR}.\rho).\phi \ .$$

□

The trick here is simply to add the cost function explicitly, as an expectation depending only on the state. Unfortunately however, that will not work for *MDP*'s with cost functions that depend on the transition as well as the state, since the transition taken is 'hidden' inside the transformer *plp* or *pgp*.

In spite of the limitations of Lem. 4.2, the expressions there tell us what we should be looking for: a more general form of expectation transformer, made by extending our existing model [20] with an explicit additive expectation. The challenge would then be to give a complete algebraic characterisation for those extended transformers, by analogy with the sublinearity that characterises our existing model.

In the next section we show that an existing extension, the 'Lamington model', has many of the characteristics we need. With it, we propose a richer model and logic for programs, one which is closer to *MDP*'s in that we are able to attach different costs to different transitions, whilst still avoiding the details of labelling.

5 The Lamington logic for *MDP*'s

The *Lamington model* for probabilistic programs, described in the appendix, was developed for a different purpose; nevertheless it turns out that is has the additive features we need for our application to *MDP*'s, and — as a bonus — we have already characterised its algebra.

Recall first that $\mathcal{R}S$ naturally accommodates varying probabilistic *transitions*. We extend this feature to varying cost-per-transition simply by adding a distinguished exit-branch to every state: like the others, it is weighted probabilistically; but, unlike the others, the contribution it makes to the overall reward is independent of the following 'reward function'. We instead introduce a unique state $\top$, so that the distinguished transitions are just the ones that terminate there.

Definition 5.1 *The Lamington relations* (following Def. A.15) Extend the state space S to $S_\top$ by adding an extra distinguished state $\top$. The *Lamington relations* are those relations r in $\mathcal{R}(S_\top)$ — over $S_\top$ rather than S — that satisfy the extra condition

$$r.\top \quad = \quad \{\overline{\top}\}$$

Thus we introduce an extra state $\top$; and we insist that a program, once in that state, remain there.[9]

We write $\mathcal{L}S$ for the Lamington relations over S.

The corresponding (minimising) Lamington transformer for a Lamington relation r is written $llp.r$, and they act over $\mathcal{E}S$ (rather than $\mathcal{E}S_\top$). □

With the Lamington model, we can now treat a larger class of *MDP*'s, although (unfortunately) there are still restrictions. We say that an *MDP* (ρ, ω) is 1-*bounded* if ω is everywhere non-negative and if

$$\omega.s.l + \sum_{s':S} \rho.s.l.s' \leq 1$$

for all states s and labels l.

To make that condition feasible, we relax the type of ρ from distributions to *sub*-distributions, thus leaving the necessary space for the added $\omega.s.l$ to 'fit in' beneath 1. Our relational spaces $\mathcal{R}S$ and $\mathcal{R}S_\top$ are similarly extended.

It turns out that Lamington programs are equivalent to 1−bounded *MDP*'s.

Definition 5.2 We define a function $_{mToL}$ which takes a *MDP* given by (ρ, ω) to a Lamington relation:

$$_{mToL}.\rho.\omega.s \quad := \quad \overline{S_\top} \cap \{l\colon L \bullet \rho.s.l + \omega.s.l \times \overline{\top}\} \cup \{\overline{\top}\} \ .$$

We take the final-state distribution (over S) given by the transition function ρ, and we add to it a transition to $\top$, weighted by the reward as given by ω; the result overall is a distribution over $S_\top$. As technicalities we exclude with $\overline{S_\top} \cap$ any putative distributions that in fact sum to more than 1, and to forestall possible emptiness we add the 'all to top' result distribution explicitly.[10] □

[9] As with our earlier definitions, there are further 'closure' conditions which we supress for simplicity. Def. A.15 includes them.

[10] Adding $\{\top\}$ is justified by the fact it would be added anyway by the closure conditions, which we have here elided. See Def. A.15.

Observe that $_{mToL}$ is well-defined even for non-1-bounded *MDP*'s, although in some cases the range may only contain $\overline{\top}$.

Theorem 5.3 If (σ, ω) is a 1-bounded *MDP*, then the least expected payoff satisfies

$$\underline{val}.\rho.\omega \quad = \quad llp.(_{mToL}.\rho.\omega) \ .$$

Proof: The hypotheses imply that $\rho.s.l + \omega.s.l \times \overline{\top} \ \in \ \overline{S_\top}$, and hence in this case $_{mToL}.\rho.\omega$ merely forgets the labels. The result then follows from simple arithmetic. □

Because 1-bounded *MDP*'s correspond to Lamington programs, they share all of the pleasant properties of programs in general, including the fact that accumulation games always have a defined limit, and that the limit is achievable via a memoryless strategy.

We now set out both accumulation games and memoryless strategies in the program model.

Definition 5.4 *Accumulation games* Let r be in $\mathcal{L}S$. The *least expected payoff* in the accumulation game r is defined to be the least fixed-point of the corresponding expectation transformer, that is

$$\mu.(llp.r) \ .$$

□

Since the transformer space is ordered, with a top- and a bottom state, least- and greatest fixed-points of monotone functions are always well-defined.

Memoryless strategies in *MDP*'s correspond to programs in which any demonic nondeterminism at any state is resolved in the same way every time that state is encountered; such programs are called 'pre-deterministic'. Their only source of demonic nondeterminism is that they might fail to terminate at all (they might 'abort', in the language of Dijkstra's weakest preconditions). The next definition gives the characteristic logical property of deterministic programs.

Definition 5.5 A program r in $\mathcal{L}S$ is said to be *pre-deterministic* if $r.s$ is a singleton set of distributions.[11]

At the level of transformers, the corresponding algebraic property is that $llp.r - llp.r.\underline{0}$ distributes addition [15, Thm. 3.5]. □

[11] We exclude any multiplicity in the set that is due only to closure conditions.

We can now prove that all 1-bounded *MDP*'s have memoryless strategies.

Theorem 5.6 The least expected reward for a 1-bounded *MDP*-accumulation game can be realised by a memoryless strategy.

Proof: From Thm. 5.3 and Def. 5.4 we can identify the least expected rewards in the two models:

$$\sum_{t:\mathbb{N}} \underline{val}\ _t.\rho.\omega \quad = \quad \mu(llp.(_{mToL}.\rho.\omega)) \ .$$

We have $_{mToL}.\rho.\omega.\underline{1} \leq \underline{1}$ since, by assumption, (ρ, ω) is 1-bounded, and so the least fixed-point exists: let that fixed point — an expectation — be ϕ. Now we take a pre-deterministic refinement r' of $_{mToL}.\rho.\omega$ such that

$$llp.r'.\phi \quad \equiv \quad llp._{mToL}.\rho.\omega.\phi \ ,$$

something that is always possible if the state space is finite [19]. Next we show that $\mu(llp.r') \equiv \mu(llp._{mToL}.\rho.\omega)$, reasoning as follows.

From monotonicity we have immediately that $\mu(llp.r') \Leftarrow \phi$. Conversely, since $e \equiv llp.r'.\phi$, we must have (by the least fixed-point property) that $\mu(llp.r') \Rrightarrow \phi$, and the result follows. □

Notice how the least fixed-point property allows us to deduce the result immediately, without having to consider first the set of memoryless strategies.

Finally we show that Thm. 5.3 applies to discounted cost problems. We consider only the case where the costs are non-negative.

Definition 5.7 *A non-negative discounted MDP* (σ, ω, β) is one for which $\omega \Leftarrow \underline{0}$. □

Any non-negative discounted *MDP* is equivalent to a 1-bounded (un-discounted) *MDP*, for calculation of least expected reward.

Lemma 5.8 Let (ρ, ω, β) be a non-negative discounted *MDP* with least expected reward ϕ; then there is a 1-bounded (un-discounted) *MDP* with least expected reward ϕ' such that $\phi = \phi' \times \sqcup\omega \times (1-\beta)$.

Proof: We construct the 1-bounded *MDP* (ρ', ω') with equivalent reward simply by 'taking up' the discount factor β in the transition probabilities (which accounts for the discounting), and by scaling the reward ω as necessary to achieve 1-boundedness.

In detail, we define ρ' to be $\beta \times \rho$, and ω' is defined $\omega/(\sqcup\omega \times (1-\beta))$. The rest is arithmetic. □

That gives us our correspondence between discounted *MDP*'s and the Lamington model.

Corollary 5.9 Any non-negative discounted *MDP* game has a memoryless strategy.
Proof: Follows from Lem. 5.8 and Thm. 5.6. □

6 Conclusions; further work

The direct connection between Markov processes and probabilistic-program semantics is only to be expected: both model a stochastic activity evolving in a series of discrete steps. Going 'forwards', the Markov approach multiplies distributions by matrices, while the program-semantics approach (Kleisli-) composes state-to-distribution functions; going 'backwards', the former multiplies expectations by matrices, and the latter (functionally-) composes expectation transformers.

In fact these four approaches correspond so well that there might not be much to choose for simplicity between them. When nondeterminism is introduced, however, the last — expectation transformers — is singled out because *it does not need to be extended*.

Markov processes must be extended to Markov *decision* processes, with their labels and strategies; and state-to-distribution functions become state-to-distribution relations. But expectation transformers retain their 'shape', merely broadening the class transformers that are considered well-formed;[12]thus many of the simpler notions applicable to the deterministic case can be carried over unchanged.

An example of that — but not our topic here — is stationarity: it is well know that a long-term 'stationary' distribution exists for any Markov process whose transition matrix satisfies certain conditions; but if those conditions are not met, the stationary distribution does not exist. McIver shows [14] that when recast as an expectation transformer the process has a stationary distribution in all cases, and considerable simplification results: the traditional conditions merely ensure that the distribution is 'deterministic'.

[12]In standard programming the transformers drop their 'disjunctivity' when nondeterminism is introduced, retaining conjunctivity; and deterministic expectation transformers are linear, becoming (only) sublinear when nondeterminism is introduced.

Our topic here has been the more general simplification that might result from treating the nondeterminism of *MDP*'s in the transformer style. In particular, the whole apparatus of state sequences, strategies and labels is 'sucked up' into the provisions that transformers make automatically for nondeterminism. Rather than seek extremal strategies — which of necessity requires that strategies themselves be modelled explicitly — we look instead at extremal fixed-points of functions. The price we pay for that is the explit modelling of the program lattice.

Whether this program-semantics approach is simpler we have yet to see. And the Lamington model, fortuitously available to accommodate additive rewards, is still limited by its being able to handle only 1-bounded processes.

Our future work is to build a model free of that restriction, using the Lamington as a guide (rather than taking it as-is), and then to elucidate its characteristic healthiness conditions. We hope that the result will be a simpler presentation of further aspects of Markov Decision Processes.

References

[1] A. Aziz, V. Singhal, F. Balarinand R.K. Brayton, and A.L. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Computer-Aided Verification, 7th Intl. Workshop*, number 939 in LNCS. Springer Verlag, 1995.

[2] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, number 1026 in LNCS, pages 499–512, December 1995.

[3] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proceedings of CONCUR99*, LNCS. Springer Verlag, 1999.

[4] E.W. Dijkstra. *A Discipline of Programming.* Prentice Hall International, Englewood Cliffs, N.J., 1976.

[5] Yishai A. Feldman and David Harel. A probabilistic dynamic logic. *J. Computing and System Sciences*, 28:193–215, 1984.

[6] J. Filar and O.J. Vrieze. *Competitive Markov Decision Processes — Theory, Algorithms, and Applications*. Springer-Verlag, 1996.

[7] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.

[8] Jifeng He, K. Seidel, and A.K. McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–192, 1997. Also available at [21].

[9] Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of 12th annual IEEE Symposium on Logic in Computer Science*, 1997.

[10] C. Jones. Probabilistic nondeterminism. Monograph ECS-LFCS-90-105, Edinburgh University, 1990. (Ph D Thesis).

[11] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.

[12] D. Kozen. A probabilistic PDL. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, New York, 1983. ACM.

[13] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[14] A.K. McIver. Stationarity in probabilistic programs. In *Proc. 17th Conf. Foundations Prog. Semantics*, Århus, May 2001.

[15] A.K. McIver and C. Morgan. Demonic, angelic and unbounded probabilistic choices in sequential programs. *Acta Informatica*, 37:329–354, 2001.

[16] Carroll Morgan and Annabelle McIver. An expectation-based model for probabilistic temporal logic. *Logic Journal of the IGPL*, 7(6):779–804, 1999.

[17] Carroll Morgan and Annabelle McIver. *pGCL*: Formal reasoning for random algorithms. *South African Computer Journal*, 22, March 1999. Also available at [21].

[18] Carroll Morgan and Annabelle McIver. Almost-certain eventualities and abstract probabilities in quantitative temporal logic. In *Proceedings CATS '01*. Elsevier, 2000.

[19] C.C. Morgan. Proof rules for probabilistic loops. In He Jifeng, John Cooke, and Peter Wallis, editors, *Proceedings of the BCS-FACS 7th Refinement Workshop*, Workshops in Computing. Springer Verlag, July 1996. `http://www.springer.co.uk/ewic/workshops/7RW`.

[20] C.C. Morgan, A.K. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, May 1996.

[21] PSG. Probabilistic Systems Group: Collected reports. `http://web.comlab.ox.ac.uk/oucl/research/areas/probs/bibliography.html`.

[22] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

[23] Colin Stirling. Local model checking games. In *CONCUR 95*, number 962 in LNCS, pages 1–11. Springer Verlag, 1995. Extended abstract.

[24] Moshe Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th Ann. ACM Symp. on Principles of Programming Languages*. ACM, January 1988. Extended abstract.

A The Lamington models

The Lamington models extend the ordinary models of probabilistic computation by placing an extra program **magic** at the top of the program-refinement lattice. Unlike the ∞-based magic program that can easily be added to our earlier probabilistic models [20, 8], the **magic** program in the Lamington models does not act as a zero for probablistic choice ${}_p\oplus$.

We begin by reviewing the earlier models, considering both unbounded and 1-bounded formulations; we use Φ for typical unbounded expectations and ϕ for typical 1-bounded expectations.

We then introduce the Lamington models and explain the way in which they relate to each other and to the earlier models.

A.1 1-bounded and 'separated' sublinearity conditions

We propose a set of sublinearity conditions that characterise transformers when restricted to 1-bounded expectations.

We know that (unbounded) transformers are characterised by[13] the 'sublinearity condition' [20, Def. 7.1].

Definition A.1 A transformer j is *sublinear* provided that for all unbounded expectations Φ_1, Φ_2 and real constants $c, c_1, c_2 \geq 0$ we have

$$j.(c_1{\times}\Phi_1 + c_2{\times}\Phi_2 \;\ominus\; \underline{c}) \quad \Leftarrow \quad c_1{\times}j.\Phi_1 + c_2{\times}j.\Phi_2 \;\ominus\; \underline{c}\;.$$

□

One of the consequences of sublinearity is 'feasibility' (see Def. A.2 (2) below), in particular that the 1-bounded subset of all expectations is closed under sublinear transformers; so it is easy to restrict our attention to 1-bounded expectations when working with sublinear transformers. The sublinearity condition itself, however, is not very convenient for the 1-bounded case, because it contains potentially non-1-bounded subexpressions (like $c_1{\times}\Phi_1 + c_2{\times}\Phi_2$ for arbitrary $c_1, c_2 \geq 0$). Our first step is to rewrite Def. A.1 into an equivalent form more suitable for 1-bounded expectations.

Easy consequences of sublinearity are the following five conditions, in which only 1-bounded expectations occur:

[13] 'Characterised by' means that the set of transformers satisfying that condition can be placed in one-to-one correspondence with the relational probabilistic programs.

Definition A.2 *Separated sublinear conditions, for the 1-bounded model* For all transformers j and 1-bounded expectations ϕ, ϕ' we have

1. Monotonicity: if $\phi \Rrightarrow \phi'$ then $j.\phi \Rrightarrow j.\phi'$.
2. Feasibility: $j.\phi \quad \Rrightarrow \quad \sqcup \phi$.
3. Scaling: $c \times j.\phi \quad \equiv \quad j.(c \times j.\phi)$ for $0 \leq c \leq 1$.
4. ${}_p\oplus$-subdistribution: $j.\phi \ {}_p\oplus \ j.\phi' \quad \Rrightarrow \quad j.(\phi \,{}_p\oplus\, \phi')$.
5. $\ominus$-subdistribution: $c{\times}j.\phi \ \ominus \ \underline{c'} \quad \Rrightarrow \quad j.(c{\times}\phi \ \ominus \ \underline{c'})$ for $0 \leq c, c'$ satisfying $c{-}c' \leq 1$.

□

It is not difficult to show the converse: the separated and restricted conditions applied to unbounded expectations imply the single Def. A.1[14]; furthermore (by scaling), a sublinear transformer's general behaviour is determined by its behaviour on 1-bounded expectations.

Note also that conditions (1,2) can be derived from the other three.

A.2 The 1-bounded Lamington transformers

We now introduce a transformer '**magic**', intended to be the 'top' of the program-refinement lattice for the 1-bounded transformers; we will see that it does not satisfy sublinearity, and so the space of transformers will have to be adjusted to accommodate it. The resulting space will be called the '1-bounded Lamington transformers'.

Definition A.3 *The **magic** transformer* For all 1-bounded expectations ϕ we define $\mathbf{magic}.\phi := \underline{1}$. □

Note that **magic** does not satisfy scaling (for example).

A similar top-of-the-lattice transformer m in the unbounded model would need to satisfy $m.\Phi := \underline{\infty}$, requiring therefore the (not difficult) extension of that model to include infinite values [17]. That m however then acts as a zero for ${}_p\oplus$ (since $\infty \,{}_p\oplus\, x = \infty$ for all $x \geq 0$ and $p > 0$), which in some

[14] Use (3) to write $j.(c_1{\times}\Phi_1 + c_2{\times}\Phi_2 \ \ominus \ \underline{c})$ as $1/e \times j.(ed \times (\Phi_1/k_1 \,{}_p\oplus\, \Phi_2/k_2) \ \ominus \ \underline{ec})$ for suitable d, e, k_1, k_2 that make everything 1-bounded and ensure $ed{-}ec \leq 1$; then use (4,5).

applications we do not want. Note however that m would satisfy sublinearity (extended with infinite arithmetic): it is thus a less ambitious extension.

But transformer **magic** does not satisfy sublinearity: take $c_1 = c_2 = 1$, $c = 0$ and $\Phi_1 \equiv \Phi_2 \equiv \underline{0}$ in Def. A.1, whence $\underline{1} \Leftarrow\!\!\!\!\Leftarrow \underline{2}$ would result. In the 'separated' form Def. A.2, however, only Conditions (2,3) fail — thus we define the Lamington conditions to be the remaining three. We call the transformers satisfying them the '1-bounded Lamington transformers':

Definition A.4 *The 1-bounded Lamington transformers* The 1-bounded Lamington transformers j take 1-bounded expectations to 1-bounded expectations, and satisfy these conditions: for all 1-bounded expectations ϕ, ϕ' we have

1. Monotonicity: if $\phi \Rrightarrow \phi'$ then $j.\phi \Rrightarrow j.\phi'$.

4. ${}_p\oplus$-subdistribution: $j.\phi \;\; {}_p\oplus \;\; j.\phi' \quad \Rrightarrow \quad j.(\phi \,{}_p\oplus\, \phi')$.

5. $\ominus$-subdistribution: $c{\times}j.\phi \;\ominus\; \underline{c'} \quad \Rrightarrow \quad j.(c{\times}\phi \ominus \underline{c'})$ for $0 \leq c, c'$ satisfies $c{-}c' \leq 1$.

□

We now look at two other Lamington models, induced from this one: an extended transformer model with explicit 'top' state $\top$, and a similarly extended relational model. All three are placed in 1-1 correspondence.

A.3 The other Lamington models

A relational model corresponding to the 1-bounded Lamington transformers must be one in which there is a place for an explicit **magic** (relational) program whose corresponding transformer has the defining property Def. A.3 above. (There can be no such program in the usual relational model [20, 8], since all transformers generated from it satisfy sublinearity.)

We construct the model going via an extended transformer space in which sublinearity has been restored: that allows us to use the fundamental theorem [20, Thm. 8.7] that puts sublinear transformer spaces in 1-1 correspondence with relational models. The state space S is extended with a special 'top' element $\top$, which in the end will be the state to which the relational **magic** program 'takes the computation':

Definition A.5 *Extended state space*

Let $S_\top := S \cup \{\top\}$ extend the original state space S.

For 1-bounded ϕ acting over S define its extension $\phi_\top$ over $S_\top$ by letting $\phi_\top.\top := 1$. For unbounded Φ acting over $S_\top$ define Φ_- to be its restriction to S. □

Our plan is now as follows. First we show that a map $(\cdot)_\top$ between 1-bounded and unbounded Lamington transformers is an injection, and that the resulting unbounded Lamington transformers are exactly those that satisfy sublinearity (over $S_\top$) together with two additional conditions.

We then show that those transformers correspond 1-1 with a subset of the probabilistic relational model [20, 8], and giving the characteristic properties of that.

Overall the result is thus a 1-1 correspondence between the three models.

A.3.1 The unbounded Lamington transformers

We begin by mapping the 1-bounded transformers into the extended space.

Definition A.6 *Unbounded Lamington transformers* The *unbounded Lamington transformers* are those transformers $j_\top$ constructed from 1-bounded Lamington transformers as follows: for 1-bounded Lamington transformer j over S, define its extension $j_\top$ acting over $S_\top$ as follows: if $\Phi.\top = 0$ then $j_\top.\Phi \equiv \underline{0}$; otherwise

$$j_\top.\Phi.\top \;:=\; \Phi.\top \text{ ; and}$$

$$\text{for } s \neq \top, \quad j_\top.\Phi.s \;:=\; \Phi.\top \times j.(\tfrac{\Phi_-}{\Phi.\top} \sqcap \underline{1}).s \ .$$

□

We show first that the unbounded Lamington transformers satisfy sublinearity, by proving the properties of Def. A.2 for transformer $j_\top$ over $S_\top$; we assume throughout all properties Def. A.4 for j over S. For brevity we may write "$c\phi$" for "$c \times \phi$" etc. In most lemmas the only non-trivial case is when the transformer(s) are not zero at $\top$ and the state is not $\top$; only where that is not so do we give the extra reasoning explicitly for $s = \top$ or $\Phi.\top = 0$.

Lemma A.7 *Monotonicity* Assume $\Phi_1 \Rrightarrow \Phi_2$; then we have

$$
\begin{array}{lll}
 & j_\top.\Phi.s & \\
= & \Phi_1.\top \times j.(\Phi_{1-}/\Phi_1.\top) \sqcap \underline{1}).s & \text{definition } j_\top \\
= & \Phi_2.\top \times (\Phi_1.\top/\Phi_2.\top) \times j.(\Phi_{1-}/\Phi_1.\top) \sqcap \underline{1}).s & \\
 & & \\
= & & j \text{ scaling; } \Phi_1.\top/\Phi_2.\top \leq 1 \\
 & \Phi_2.\top \times j.((\Phi_1.\top/\Phi_2.\top) \times (\Phi_{1-}/\Phi_1.\top) \sqcap \underline{1})).s & \\
 & & \\
= & \Phi_2.\top \times j.(\Phi_{2-}/\Phi_2.\top) \sqcap \underline{1}).s & j \text{ monotonic} \\
= & j_\top.(\Phi_2).s\ . & \text{definition } j_\top
\end{array}
$$

□

Lemma A.8 *Feasibility* This follows from scaling and $\ominus$-subdistribution (below) [20, Lem. 7.4]. (It does not hold in the Lamington model because Def. A.4 does not include scaling.) □

Lemma A.9 *Scaling*

$$
\begin{array}{lll}
 & j_\top.(c\Phi).s & \\
= & (c\Phi).\top \times j.(\frac{c\Phi_-}{c\Phi_\top} \sqcap \underline{1}).s & \text{definition } j_\top \\
= & c \times \Phi.\top \times j.(\frac{\Phi_-}{\Phi_\top} \sqcap \underline{1}).s & \\
= & c \times j_\top.\Phi.s\ . & \text{definition } j_\top
\end{array}
$$

Note we do not require $c \leq 1$. □

Lemma A.10 *${}_p\oplus$-subdistribution*

$$
\begin{array}{lll}
 & j_\top.(\Phi_1 \,{}_p\oplus\, \Phi_2).s & \\
= & j_\top.(c \times (\Phi_1/\Phi_1.\top \,{}_r\oplus\, \Phi_2/\Phi_2.\top)).s & \text{for some } 0 \leq c \text{ and } 0 \leq r \leq 1 \\
= & c \times j_\top.(\Phi_1/\Phi_1.\top \,{}_r\oplus\, \Phi_2/\Phi_2.\top).s & j_\top \text{ scaling} \\
 & & \\
= & & \text{definition } j_\top;\ (\Phi_1/\Phi_1.\top \,{}_r\oplus\, \Phi_2/\Phi_2.\top).\top = 1 \\
 & c \times j.((\Phi_1/\Phi_1.\top \,{}_r\oplus\, \Phi_2/\Phi_2.\top) \sqcap \underline{1}).s & \\
 & & \\
\geq & c \times j.((\Phi_1/\Phi_1.\top \sqcap \underline{1}) \,{}_r\oplus\, (\Phi_2/\Phi_2.\top \sqcap \underline{1})).s & \\
 & & \\
\geq & & j \text{ is } {}_p\oplus\text{-subdistributive} \\
 & c \times (j.(\Phi_1/\Phi_1.\top \sqcap \underline{1}).s \;\; {}_r\oplus \;\; j.(\Phi_2/\Phi_2.\top \sqcap \underline{1}).s) &
\end{array}
$$

$$
\begin{array}{lll}
= & c \times (j_\top.\Phi_1.s/\Phi_1.\top \ {}_r\!\oplus \ j_\top.\Phi_2.s/\Phi_2.\top)) & \text{definition } j_\top \\
= & j_\top.\Phi_1.s \ {}_p\!\oplus \ j_\top.\Phi_2.s \ . & \text{choice of } c, r
\end{array}
$$

If *wlog* $\Phi_1.\top \neq 0$ but $\Phi_2.\top = 0$, we reason

$$
\begin{array}{lll}
 & j_\top.(\Phi_1 \ {}_p\!\oplus \Phi_2).s & \\
\geq & j_\top.(p\Phi_1).s & j_\top \text{ monotonic} \\
= & p \times j_\top.\Phi_1.s & j_\top \text{ scaling} \\
= & j_\top.\Phi_1.s \ {}_p\!\oplus j_\top.\Phi_2.s \ . & \Phi_2.\top = 0
\end{array}
$$

□

Lemma A.11 *$\ominus$-subdistribution*

$$
\begin{array}{lll}
 & j_\top.(c\Phi \ \ominus \ \underline{c'}).s & \\
 & & \\
= & & \text{definition } j_\top\text{; define } d := (c\times\Phi \ominus \underline{c'}).\top\text{; see below for } d = 0 \\
 & d \times j.((c\Phi_-/d \ominus \underline{c'/d}) \sqcap \underline{1}).s & \\
 & & \\
= & & \text{define } e := \Phi.\top\text{; see below for } e = 0 \\
 & d \times j.(((ce/d)\Phi_-/e \ominus \underline{c'/d}) \sqcap \underline{1}).s & \\
 & & \\
\geq & & j \text{ monotonic; arithmetic and } ce/d - c'/d \leq 1 \text{ (see below)} \\
 & d \times j.((ce/d)(\Phi_-/e \sqcap \underline{1}) \ominus \underline{c'/d}).s & \\
 & & \\
\geq & & j \text{ subdistributes } \ominus\text{; } ce/d - c'/d \leq 1 \\
 & d \times ((ce/d)j.(\Phi_-/e \sqcap \underline{1}) \ominus \underline{c'/d}).s & \\
 & & \\
= & (ce)j.(\Phi_-/e \sqcap \underline{1}).s \ominus c' & \\
= & c(j_\top.\Phi.s) \ominus c' \ . & \text{definition } j_\top, e
\end{array}
$$

Note that from the definitions we have $d = ce - c' \sqcup 0$; that gives $ce/d - c'/d \leq 1$ immediately, and when $e = 0$ it gives $d = 0$, leaving that last as the only special case.

When $d = 0$ the *lhs* is 0 by definition of $j_\top$; the *rhs* is 0 also since, directly from the definition of $j_\top$ we have $j_\top.\Phi.s \leq \Phi.\top = e$. □

That shows that the extended transformer $j_\top$ satisfies sublinearity. We now mention the two extra properties explicitly.

Lemma A.12 *Extra unbounded Lamington properties* For any $j_\top$ and Φ we have

1. $\top$-*preserving* $\quad j_\top.\Phi.\top = \Phi.\top$; and
2. $\top$-*capped* $\quad j_\top.\Phi \equiv j_\top.(\Phi \sqcap \underline{\Phi.\top})$.

Proof: Immediate from the definition of $j_\top$. □

To complete the correspondence we construct an inverse $(\cdot)_-$ to $(\cdot)_\top$, going from the unbounded back to the 1-bounded model.

Definition A.13 For unbounded Lamington transformer k over $S_\top$, define 1-bounded Lamington transformer over S by

$$k_-.\phi \quad := \quad (k.\phi_\top)_- \ .$$

□

We conclude by showing that the two maps are inverses:

Lemma A.14 *1-1 correspondence between 1-bounded and unbounded Lamington models*

1. For 1-bounded Lamington j we have $(j_\top)_- = j$.
2. For unbounded Lamington k we have $(k_-)_\top = k$.

Proof: The proof of (1) is immediate from the definitions. For (2) we reason for cases $\Phi.\top \neq 0$ and $s \neq \top$

$$\begin{array}{lll}
 & (k_-)_\top.\Phi.s & \\
= & \Phi.\top \times k_-.(\frac{\Phi_-}{\Phi.\top} \sqcap \underline{1}) & \text{Definition } (\cdot)_\top \\
= & \Phi.\top \times k.((\frac{\Phi_-}{\Phi.\top} \sqcap \underline{1})_\top) & \text{Definition } (\cdot)_- \\
= & \Phi.\top \times k.((\Phi \sqcap \underline{\Phi.\top})/\Phi.\top) & \text{arithmetic} \\
= & k.(\Phi \sqcap \underline{\Phi.\top}) & k \text{ scaling} \\
= & k.\Phi \ . & k \text{ is } \top\text{-capped, Lem. A.12(2)}
\end{array}$$

The other cases are straightforward. □

A.3.2 The Lamington relations

Because the unbounded Lamington transformers are sublinear, we have a relational representation for them: the relational representation guaranteed by [20, Lem. 8.6] is a function from states (S extended with $\top$, in this case) to sets of sub-distributions[15] over states; since we are working with a finite S the distributions are discrete. The conditions imposed on that relational model are that the sets of distributions are non-empty, up-closed, convex-closed and Cauchy-closed.[16]

The specific origin of our transformers here — as extensions of Lamington transformers over the (non-extended) S — results in two further properties of their corresponding relations: they are '$\top$-preserving' and '$\top$-up-closed'.

A relation r is $\top$-*preserving* if $r.\{\top\} = \{\overline{\top}\}$, where for state s the distribution $\overline{s}$ is the point distribution over s. Thus a $\top$-preserving relation takes $\top$ only to $\top$: put operationally, it means that 'once the program reaches the $\top$ state, it stays there'.

Our relations are $\top$-preserving because of the $\top$-preserving condition on the transformers, which in particular gives $j_\top.[s = \top].\top = 1$, where in general the function $[b]$ maps Boolean b to the expectation 1 $\underline{\text{if}}$ b $\underline{\text{else}}$ 0 — thus we have that the probability that $j_\top$ takes $\top$ to $\top$ is 1.

The other condition extends ordinary up-closure: for sub-distributions d, d' we say that $d \sqsubseteq d'$ whenever $d.s \leq d'.s$ for all states (including $\top$ in this case). (Note that proper- rather than sub-distributions are therefore $\sqsubseteq$-maximal.) Informally, to move $\sqsubseteq$-upwards one is allowed to 'steal' probability from the implicit non-termination case and 'give it' to any proper state; and that (smaller chance of non-termination) is considered to be 'better' in the usual sense of refinement.

Informally, $\top$-up-closure allows one as well to steal probability from any

[15] A *sub-distribution* sums to no more than 1; the deficit if any is the probability of non-termination.

[16] The representing relation is given by [20, Def. 8.1]

$$\begin{array}{rcl} rp.j_\top.s & := & \{d \mid (\forall \Phi \cdot \int_d \Phi \geq j_\top.\Phi.s)\} \\ & = & \text{when } s \neq \top \\ & & \{d \mid (\forall \phi \cdot d.\top + \int_{d\_} \phi \geq j.\phi.s)\} \ , \end{array}$$

where $d\_$ is the distribution d restricted to S and expectations ϕ as usual are 1-bounded and ranging over S.

state and give it to $\top$. Thus we say that $d \sqsubseteq_\top d'$ whenever for all subsets S' of proper states we have $d.S' + d.\top \leq d'.S' + d'.\top$, and a set of distributions is $\top$-*up-closed* if it is up-closed under $\sqsubseteq_\top$. Again the notion of refinement suggests that a higher probability of reaching $\top$ is an improvement.

Our relations are $\top$-up-closed because of the $\top$-clipping property of the transformers, giving $\Phi.\top \geq \Phi.s$ for all Φ that we are actually interested in. So to steal from s and give to $\top$ in a distribution d can only increase the expectation $\int_d \Phi$, which results in the closure of the relational model.

We note that these conditions follow naturally from the general constructions of Clare Jones [10] if we work over the base domain $\bot \sqsubseteq S \sqsubseteq \top$ rather than just our usual $\bot \sqsubseteq S$.

Thus we define

Definition A.15 *The Lamington relations* The Lamington relations are those relations r in the probabilistic relational model [20, 8] over $S_\top$ that satisfy the two extra conditions

1. $\top$-*preserving* $\quad r.\{\top\} = \{\overline{\top}\}$; and
2. $\top$-*up-closed* If $d \in r.s$ and $d \sqsubseteq_\top d'$ then $d' \in r.s$ also.

□

To place the unbounded Lamington transformers in 1-1 correspondence with the Lamington relations we rely mainly on the 1-1 correspondence guaranteed by sublinearity; in addition we need only show that the conditions of Lem. A.12 and Def. A.15 map to each other in that correspondence. The implication from Lem. A.12 to Def. A.15 has been discussed above; and that a $\top$-preserving relation yields a $\top$-preserving transformer is straightforward.

Thus we are left only with the fact that a $\top$-up-closed relation yields a $\top$-capped transformer; we sketch the proof of that here. Fix initial state s and suppose $\top$-up-closed relation r yields transformer k; we consider $k.\Phi.s$ for some expectation Φ where *wlog* we let $\Phi.s' \geq \Phi.\top$ at some (final) state s'. Let d be any of the distributions on the boundary of $r.s$ that determine the value $k.\Phi.s$: if $d.s' > 0$ then we can construct a d', by moving all probability in d from s' to $\top$, so that $d'.s', d'.\top = 0, d.\top + d.s'$ and $\int_{d'} \Phi \leq \int_d \Phi$ — which means that d' is also a determining distribution for $k.\Phi.s$.

Thus the determining distributions of unbounded expectations can be taken to be 0-valued at all states in which the expectation is at least its value at $\top$; and that means the effect of the transformer cannot be sensitive to the uncapped values of that expectation.

A.4 Summary

In this appendix we have considered six models of probabilistic computation.

The first three are the 'ordinary' models, without **magic**: the unbounded transformer model, the 1-bounded transformer model and the relational model. The first and third have been much discussed, and shown to correspond, elsewhere [20]; our contribution here was to set out the 1-bounded model, to show it 1-1 corresponds with the other two, and to rewrite the sublinearity condition in a form convenient for it.

The other three models are the Lamington models, extending the ordinary models with **magic**. The first was 1-bounded transformers, derived from the ordinary 1-bounded transformers by relaxing their characteristic conditions so that **magic** could be accommodated. We then showed 1-1 correspondence of that model with an unbounded transformer model and with a relational model, both of those acting over a state space extended with an extra 'top' element $\top$ and being restrictions of the ordinary (but $\top$-extended) models there.