A New Proof Rule for Almost-Sure Termination

ANNABELLE MCIVER, Macquarie University, Australia
CARROLL MORGAN, University of New South Wales, Australia and Data61, CSIRO, Australia
BENJAMIN LUCIEN KAMINSKI, RWTH Aachen University, Germany and UCL, UK
JOOST-PIETER KATOEN, RWTH Aachen University, Germany and IST, Austria

We present a new proof rule for proving almost-sure termination of probabilistic programs, including those that contain demonic non-determinism.

An important question for a probabilistic program is whether the probability mass of all its diverging runs is zero, that is that it terminates "almost surely". Proving that can be hard, and this paper presents a new method for doing so. It applies directly to the program's source code, even if the program contains demonic choice.

Like others, we use variant functions (a.k.a. "super-martingales") that are real-valued and decrease randomly on each loop iteration; but our key innovation is that the amount as well as the probability of the decrease are *parametric*. We prove the soundness of the new rule, indicate where its applicability goes beyond existing rules, and explain its connection to classical results on denumerable (non-demonic) Markov chains.

CCS Concepts: • **Theory of computation → Program verification**; *Probabilistic computation*; *Axiomatic semantics*;

Additional Key Words and Phrases: Almost-sure termination, demonic non-determinism, program logic pGCL.

ACM Reference Format:
Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. 2018. A New Proof Rule for Almost-Sure Termination. *Proc. ACM Program. Lang.* 2, POPL, Article 33 (January 2018), 28 pages. https://doi.org/10.1145/3158121

1 INTRODUCTION

This paper concerns termination proofs for sequential, imperative *probabilistic* programs, i.e. those that, in addition to the usual constructs, include a binary operator for probabilistic choice. Writing "standard" to mean "non-probabilistic", we recall that the standard technique for loop termination is to find an *integer-valued* function over the program's state space, a "variant", that satisfies the "progress" condition that each iteration is guaranteed to decrease the variant strictly and further that the loop guard and invariant imply that the variant is bounded below by a constant (typically zero). Thus it cannot continually decrease without eventually making the guard false; and so existence of such a variant implies the loop's termination.

For probabilistic programs, the definition of loop termination is often weakened to "almost-sure termination", or "termination with probability one", by which is meant that the probability of the loop's iterating forever is zero. For example if you flip a fair coin repeatedly until you get heads,

Authors' addresses: Annabelle McIver, Macquarie University, Australia, annabelle.mciver@mq.edu.au; Carroll Morgan, University of New South Wales, Australia , Data61, CSIRO, Australia, carroll.morgan@unsw.edu.au; Benjamin Lucien Kaminski, RWTH Aachen University, Germany , UCL, UK, benjamin.kaminski@informatik.rwth-aachen.de; Joost-Pieter Katoen, RWTH Aachen University, Germany , IST, Austria, katoen@cs.rwth-aachen.de.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
2475-1421/2018/1-ART33
https://doi.org/10.1145/3158121

you will eventually stop — the probability of flipping tails forever is zero. We will write *AS* for "almost sure" and *AST* for "almost-sure termination" or "almost-surely terminating".

But the standard variant rule we mentioned above is too weak for *AST* in general. Write $Com \;{}_p\!\oplus\; Com'$ for choice of Com, Com' with probability p, $1{-}p$ resp. and consider the *AST* program

$$x \coloneqq 1; \quad \texttt{while}\,(x{\neq}0)\,\{\; x \coloneqq (x{+}1) \bmod 3 \;{}_{1/2}\!\oplus\; x \coloneqq (x{-}1) \bmod 3 \;\}\,. \tag{1}$$

It has no standard variant, because that variant would have to be decreased strictly by both updates to x. Also the simple *AST* program

$$\text{1dSRW:} \qquad \texttt{while}\,(x{\neq}0)\,\{\; x \coloneqq x{+}1 \;{}_{1/2}\!\oplus\; x \coloneqq x{-}1 \;\}\;, \tag{2}$$

the *symmetric random walk* over integers x, is beyond the reach of the standard rule.

Thus we need *AST*-rules for properly probabilistic programs, and indeed many exist already. One such, designed to be as close as possible to the standard rule, is that an integer-valued variant must be bounded *above* as well as below, and its strict decrease need only occur with *non-zero probability* on each iteration, i.e. not necessarily every time [McIver and Morgan 2005, Lem.2.7.1]. [1] That rule suffices for Program (1) above, with variant x and upper bound 2; but still it does not suffice for Program (2).

The 1dSRW is however an elementary Markov process, and it is frustrating that a simple termination rule like the above (and some others' rules too) cannot deal with its *AST*. This (and other examples) has led to many variations in the design of *AST*-rules, a competition in which the rules' assumptions are weakened as much as one dares, to increase their applicability beyond what one's colleagues can do; and yet of course the assumptions must not be weakened so much that the rule becomes unsound. This is our first principal **Theme (A)** — the power of *AST*-rules.

A second **Theme (B)** in the design of *AST*-rules is their applicability at the source level (of program texts), i.e. whether they are expressible and provable in a (probabilistic) program logic without "descending into the model". We discuss that practical issue in §2 and App. E.3 — it is important e.g. for enabling theorem proving.

Finally, a third **Theme (C)** is the characterisation of the kinds of iteration for which a given rule is guaranteed to work, i.e. a completeness result stating for which *AST* programs a variant is guaranteed to exist, even if it is hard to find. Typical characterisations are "over a finite state space" [Hart et al. 1983],[McIver and Morgan 2005, Lem. 7.6.1] or "with finite expected time to termination" [Ferrer Fioriti and Hermanns 2015].

The contribution of this paper is to cover those three themes. We give a novel rule for *AST*, one that: (A) proves almost-sure termination in some cases that lie beyond what some other rules can do; (B) is applicable directly at the source level to probabilistic programs *even if they include demonic choice*, for which we give examples; and (C) is supported by mathematical results from pre- computer-science days that even give some limited completeness criteria. In particular, one of those classical works shows that our new rule must work for the *two-dimensional* random walk: a variant is guaranteed to exist, and to satisfy all our criteria. That guarantee notwithstanding, we have yet to find a 2dSRW-variant in closed form.

2 OVERVIEW

Expressed very informally, the new rule is this:

> Find a non-negative *real-valued* variant function V of the state such that: (1) iteration cannot increase V's expected value; (2) on each iteration the actual value v of V must

[1] Over an infinite state space, the second condition becomes "with some probability bounded away from zero".

decrease by at least $d(v)$ with probability at least $p(v)$ for some fixed non-increasing strictly positive real-valued functions d, p; [2] and (3) iteration must cease if $v{=}0$.

The formal statement of the rule, and a more detailed but still informal explanation, is given in §4.2.

Section 3 gives notation, and a brief summary of the programming logic we use. Section 4.3 uses that logic to prove the new rule rigorously; thus we do not reason about transition systems directly in our proof. Instead we rely on the logic's *being valid* for transition systems (e.g. valid for Markov decision processes), for the following two reasons:

Recall Theme (A) — The programming logic we use –its theorems to which we appeal– are valid even for programs that contain demonic choice. And so our result is valid for demonic choice as well. (In §8.1 we discuss the degree of demonic choice that is permitted.)

Recall Theme (B) — Expressing the termination rule in terms of a programming logic means that it can be applied to source code directly and that theorems can be (machine-) proved about it: there is no need to translate the program first into a transition system or any other formalism. The logic we use is a probabilistic generalisation of (standard) Hoare/Dijkstra logic [Dijkstra 1976], due to Kozen [1985] and later extended by Morgan et al. [1996] and McIver and Morgan [2005] to (re-)incorporate demonic choice.

Section 5 carefully applies the rule to several small examples, illustrating its power and the logical manipulations it induces. Section 6 explores the classical literature on *AST*. Section 7 examines other contemporary *AST* rules. Section 8 treats some theoretical aspects and limitations.

3 PRELIMINARIES

3.1 Programming Language and Semantics

pGCL is a simple imperative programming language based on Dijkstra's GCL 1976 but with an additional operator of binary probabilistic choice ${}_p\oplus$ introduced by Kozen [1985] and extended by Morgan et al. [1996] and McIver and Morgan [2005] to co-exist with demonic choice. Its forward, operational model is functions from states to sets of discrete distributions on states, where the sets represent demonic nondeterminism if it is present: this is essentially Markov decision processes, but also probabilistic/demonic transition systems. (§8.1 describes some of the conditions imposed on the "demonic" sets.) Its backwards, logical model is functions from so-called "post-expectations" to "pre-expectations", non-negative real valued functions on the state that generalise the postconditions and preconditions of Hoare/Dijkstra [Hoare 1969] that are Boolean functions on the state: that innovation, and the original link between the forwards and backwards semantics, due to Kozen [1985] but using our terminology here, is that $A = \mathsf{wp}.Com.B$, for pGCL program Com and post-expectation B, means that pre-expectation A is a function that gives for every initial state the expected value of B in the final distribution reached by executing Com. The demonic generalisation of that [McIver and Morgan 2005; Morgan et al. 1996] is that A gives the *infimum* over all possible final distributions of B's expected value. Both of these generalise the "standard" Boolean interpretation exactly if false is interpreted as zero, true as one and implication as ($\leq$) (and therefore conjunction as infimum).

pGCL's weakest pre-expectation logic, like Dijkstra's weakest precondition logic, is designed to be applied at the source-code level of programs, as the case studies in §5 illustrate. Its theorems etc. are also expressed at the source-code level, but apply of course to whatever semantics into which the logic is (validly) interpreted.

We now set out more precisely the framework in which we operate. Let Σ be the set of program states. We call a subset G of Σ a *predicate*, equivalently a function from Σ to the Booleans. If Σ is

[2] As §8.2 explains, functions d, p must have those properties for *all* positive reals, not only the V's that are reachable.

Table 1. Rules for the expectation-transformer wp.

C	$\mathbf{wp}\,.\,C\,.\,f$
`skip`	f
$x \coloneqq e$	$f[x/e]$
`if` $(G)\,\{C_1\}$ `else` $\{C_2\}$	$[G]\cdot \mathsf{wp}\,.\,C_1\,.\,f + [\neg G]\cdot \mathsf{wp}\,.\,C_2\,.\,f$
$\{C_1\}\ {}_p\!\oplus\ \{C_2\}$	$p\cdot \mathsf{wp}\,.\,C_1\,.\,f + (1-p)\cdot \mathsf{wp}\,.\,C_2\,.\,f$
$\{C_1\}\ \Box\ \{C_2\}$	$\min\{\mathsf{wp}\,.\,C_1\,.\,f,\ \mathsf{wp}\,.\,C_2\,.\,f\}$
$C_1;\ C_2$	$\mathsf{wp}\,.\,C_1\,.\,\big(\mathsf{wp}\,.\,C_2\,.\,f\big)$
`while` $(G)\,\{C'\}$	$\mathsf{lfp}\,X.\ [\neg G]\cdot f + [G]\cdot \mathsf{wp}\,.\,C'\,.\,X$

In the table above C is a pGCL program, and f is an expectation. The notation $f[x/e]$ is function f overridden at argument x by the value e. A period "." denotes (Curried) function application, so that for example $\mathsf{wp}\,.\,C_1\,.\,f$ is semantic-function wp applied to the syntax C_1; the resulting transformer is then applied to the "post-expectation" f. A centred dot is multiplication, either of scalars or of an expectation by a scalar.
In ${}_p\!\oplus$ the probability p can be an expression in the program variables (equivalently a $[0,1]$-valued function of Σ). Often however it is a constant.
The operator $\Box$ is demonic choice.

the Cartesian product of named-variable types, we can describe functions on Σ as expressions in which those variables appear free, and predicates are then Boolean-valued expressions.

We use Iverson bracket notation $[G]$ to denote the *indicator function* of a predicate G, that is 1 on those states where G holds and 0 otherwise.

An *expectation* is a random variable that maps program states to non-negative reals:

Definition 3.1 (Expectations [McIver and Morgan 2005]*).* The set of expectations on Σ, denoted by $\mathbb{E}$, is defined as $\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}\}$. We say that f is *bounded* iff there exists a (non-negative) real b such that $f(\sigma) \leq b$ for all states σ. The natural complete partial order $\leq$ on $\mathbb{E}$ is obtained by pointwise lifting, that is

$$f_1 \leq f_2 \qquad \text{iff} \qquad \forall \sigma \in \Sigma\colon \quad f_1(\sigma) \leq f_2(\sigma)\ . \qquad \triangle$$

Thus Iverson brackets $[-]$ map predicates to expectations, and $(\Rightarrow)$ to $(\leq)$ similarly — that is, we have $[A] \leq [B]$ just when $A \Rightarrow B$.

Following Kozen [1985], here we are are based on Dijkstra's guarded-command language GCL [Dijkstra 1976] but extended with a probabilistic-choice operator ${}_p\!\oplus$ between program (fragments) that chooses its left operand with probability p (and its right complementarily). Beyond Kozen however, we use pGCL where demonic choice is *retained* [McIver and Morgan 2005; Morgan et al. 1996] — i.e. pGCL contains *both* probabilistic- and demonic choice. The syntax of pGCL is given in Table 1, and its semantics of *expectation transformers*, the generalisation of predicate transformers, is defined as follows:

Definition 3.2 (The wp*-Transformer* [McIver and Morgan 2005]*).* The weakest pre-expectation transformer semantic function $\mathsf{wp}\colon \text{pGCL} \to (\mathbb{E} \to \mathbb{E})$ is defined in Table 1 by induction on all pGCL programs. $\triangle$

If f is an expectation on the *final* state, then $\mathsf{wp}\,.\,Com\,.\,f$ is an expectation on the *initial* state: thus $\mathsf{wp}\,.\,Com\,.\,f\,.\,\sigma$ is the infimum, over all distributions of final states that Com can reach from σ, of the expected value of f on each final distribution: there will be more than one just when Com contains demonic choice. In the special case where f is $[B]$ for predicate B, that value is thus the least guaranteed probability with which Com from σ will reach a final state satisfying B.

The natural connection between the standard world of predicate transformers (Dijkstra) and the probabilistic expectation transformers (Kozen/pGCL) is the indicator function: for example $[\mathit{false}]$ is 0 and $[\mathit{true}]$ is 1, [3] and the predicate implication $A \Rightarrow B$ is equivalent to the expectation inequality $[A] \leq [B]$. The standard $A \Rightarrow \mathsf{wp}\,.\,Com\,.\,B$, using standard wp and program Com (i.e. without probabilistic choice in Com), becomes $[A] \leq \mathsf{wp}\,.\,Com\,.\,[B]$ when using the wp we adopt here. Finally, the idiom

$$p \cdot [A] \qquad \leq \qquad \mathsf{wp}\,.\,Com\,.\,[B]\,, \tag{3}$$

where "$\cdot$" is real-valued multiplication (pointwise lifted if necessary), means "with probability at least p the program Com will take an initial state satisfying A to a final state satisfying B", where p is a $[0,1]$-valued expression on (or equivalently a function of) the program state: in most cases however p is constant. This is because if the initial state σ does not satisfy A, i.e. $A(\sigma)$ is *false*, then the *lhs* of (3) is zero so that the inequality is trivially true; and if σ does satisfy A then the *lhs* is $p \cdot 1 = p$ (or $p(\sigma)$ more generally) and the *rhs* is the least guaranteed probability of reaching B, because the expected value of $[B]$ over a distribution is the probability that distribution assigns to B. (The "least" is, again, because of possible demonic nondeterminism.)

There are many properties of pGCL's probabilistic wp that are analogues of wp for standard programs; but one that is *not* an analogue is "scaling" [McIver and Morgan 2005, Def. 1.6.2], an intrinsically numeric property whose justification rests ultimately on the distribution of multiplication through expected value from elementary probability theory. For us it is that for all commands Com, post-expectations $Post$ and non-negative reals c we have

$$\mathsf{wp}\,.\,Com\,.\,(c \cdot Post) \qquad = \qquad c \cdot (\mathsf{wp}\,.\,Com\,.\,Post)\,. \tag{4}$$

We use it in the proof of Thm. 4.1 below. (See also App. E.2.)

3.2 Probabilistic Invariants, Variants, and Termination with Probability 1

With the above correspondence, the following probabilistic analogues of standard termination and invariants are natural.

Definition 3.3 (Probabilistic Invariants [McIver and Morgan 2005, p. 39, Definition 2.2.1]*).* Let $Guard$ be a predicate, a loop guard, and Com be a pGCL program, a loop body. Then bounded expectation Inv is a *probabilistic invariant* of the loop $\texttt{while}\,(Guard)\,\{Com\}$ just when

$$[Guard] \cdot Inv \qquad \leq \qquad \mathsf{wp}\,.\,Com\,.\,Inv\,. \tag{5}$$

In this case we say that Inv is *preserved* by each iteration of $\texttt{while}\,(Guard)\,\{Com\}$. [4] △

When some predicate Inv' is such that $Inv = [Inv']$ is a probabilistic invariant, we can equivalently say that Inv' itself is a *standard invariant* (predicate). [5]

[3] We will blur the distinction between Booleans and constant predicates, so that *false* is just as well the predicate that holds for no state. The same applies to reals and constant expectations.

[4] If (real valued) expectation Inv were equal to $[Inv']$ for some predicate Inv', we'd have $[Guard \wedge Inv'] \leq \mathsf{wp}\,.\,Com\,.\,[Inv']$, exactly the standard meaning of "preserves Inv'".

[5] For any standard program Com, i.e. without probabilistic choice, Dijkstra's GCL judgement $Inv \Rightarrow \mathsf{wp}\,.\,Com\,.\,Inv$ is equivalent to our pGCL judgement $[Inv] \leq \mathsf{wp}\,.\,Com\,.\,[Inv]$ for any predicate Inv.

In §1 we recalled that the standard method of proving (standard) loop termination is to find an integer-valued variant function *VInt* on the state such that the loop's guard (and the invariant, if one is given) imply that $VInt \geq 0$ and that *VInt* strictly decreases on each iteration. A probabilistic analogue of loop termination is "terminates with probability one", i.e. terminates almost-surely, and one (of many) probabilistic analogue(s) of the standard loop-termination rule is the following:

THEOREM 3.4 (VARIANT RULE FOR LOOPS (EXISTING: [McIver and Morgan 2005, p. 55, Lemma 2.7.1])). *Let Inv, Guard* $\subseteq \Sigma$ *be predicates; let VInt*$: \Sigma \to \mathbb{Z}$ *be an integer-valued function on the state space; let Low, High be fixed integers; let* $0 < \varepsilon \leq 1$ *be a fixed strictly positive probability that bounds away from zero the probability that VInt decreases; and let Com be a* pGCL *program. Then the three conditions*

(i) Inv is a standard invariant (equiv. $[Inv]$ *an invariant) of* $\texttt{while}\,(Guard)\,\{Com\}$*, and*
(ii) $Guard \wedge Inv \Rightarrow Low < VInt \leq High$*, and* [6]
(iii) for any constant integer N we have $\varepsilon \cdot [Guard \wedge Inv \wedge VInt{=}N] \;\leq\; \texttt{wp.}Com.[VInt{<}N]$,

when taken all together, imply $[Inv] \leq \texttt{wp.while}\,(Guard)\,\{Com\}\texttt{.}\mathbf{1}$*, that from any initial state satisfying Inv the loop terminates AS.*

The "*for any integer N*" in (iii) above is the usual Hoare-logic technique for capturing an expression's initial value (in this case *VInt*'s) for use in the postcondition: we can write "$VInt{<}N$" there for "the current value *VInt*, here in the final state, is strictly less than the value *N* it had in the initial state." [7] Recalling (3), we see that assumption (iii) thus reads

> On every iteration *Com* of the loop the variant *VInt* is guaranteed to decrease strictly with probability at least some (fixed) strictly positive ε .

The probabilistic variant rule above differs from the standard rule in two essential respects: the probabilistic variant must be bounded *above* as well as below (which tends to make the rule weaker); and the decrease need not be certain, rather only bounded away from zero (which tends to make the rule stronger). Although this rule does have wide applicability [McIver and Morgan 2005, Chp. 3], it nevertheless is not sufficient for example to show *AST* of the symmetric random walk, Program (2). [8]

The advance incorporated in our new rule, as explained in the next section, is to *strengthen Thm. 3.4 in three ways*: (1) we remove the need for an upper bound on the variant; (2) we allow the probability ε to vary; and (3) we allow the variant to be real-valued. (Thm. 3.4 is itself used as a lemma in the proof of soundness of the new rule.)

We will need the following theorem, a probabilistic analogue of the standard technique that partial correctness plus termination gives total correctness, and with similar significance: proving "only" that a standard loop terminates certainly indeed does not necessarily give information about the loop's efficiency; but the termination proof is still an essential prerequisite for other proofs about the loop's functional correctness. The same applies in the probabilistic case.

[6] The original rule [McIver and Morgan 2005, Lem. 2.7.1] had $Low \leq VInt < High$. We make this inessential change for later neatness.
[7] In greater detail: if the universally quantified *N* is instantiated to anything other than *VInt*'s initial value then the left-hand side of (iii) is zero, satisfying the inequality trivially since the right-hand side is non-negative by definition of expectations.
[8] Any variant that works for [McIver and Morgan 2005, p. 55, Lemma 2.7.1] must be bounded above and -below, and integer-valued. And it must be able (with some non-zero probability) to decrease strictly on each step. If its bounds were say *L*, *H*, then it must therefore be able to terminate from *anywhere* in no more than $H{-}L$ steps, a fixed and finite number. But (2) does not have that property.

Theorem 3.5 (Almost-sure termination for probabilistic loops (existing: [McIver and Morgan 2005, p. 43, Lemma 2.4.1, Case 2.])). *Let Term satisfy* $[Term] \leq \mathtt{wp.while}\,(Guard)\,\{Com\}\mathtt{.1}$, *that is that from any initial state satisfying Term the loop terminates AS (termination), and let* bounded *expectation Sub be preserved by Com whenever Guard holds, that is a probabilistic invariant of* $\mathtt{while}\,(Guard)\,\{Com\}$ *(partial correctness). Then*

$$[Term]\cdot Sub \quad \leq \quad \mathtt{wp.while}\,(Guard)\,\{Com\}\mathtt{.}([\neg Guard]\cdot Sub)\,. \qquad \text{(total correctness)}$$

The intuitive import of this theorem is that if bounded *Sub* is a probabilistic invariant preserved by each iteration of the loop body, then also the whole loop "preserves" *Sub* from any state where the loop's termination is *AS*. This holds even if *Com* contains demonic choice.

Bounding *Sub* is required by [McIver and Morgan 2005], where Thm. 3.5 is found, and it is necessary here (§8.4).

4 A NEW PROOF RULE FOR ALMOST-SURE TERMINATION

4.1 Martingales

Important for us in extending the *AST* rule is reasoning about "sub- and super-martingales".

A *martingale* is a sequence of random variables for which the expected value of each random variable next in the sequence is equal to the current value (irrespective of any earlier values). A *super*-martingale is more general: the current value may be larger than the expected subsequent value; and a *sub*-martingale is the complementary generalisation. In probabilistic programs, as we treat them here, such a sequence of random variables is some expectation evaluated over the succession of program states as a loop executes, and an exact/super/sub -martingale is an expectation whose value at the beginning of an iteration (a single state) is equal-to/no-less-than/no-more-than its expected value at the end of that iteration.

A trivial example of a sub-martingale is the invariant predicate of a loop in standard programming, provided we interpret *false*$\leq$*true*, for if the invariant is true at the beginning of the loop body it must be true at the end — provided the loop guard is true. More generally in Def. 3.3 above we defined a probabilistic invariant, and at (5) there we see that it is a sub-martingale, again provided the loop guard holds. (If the loop guard does not hold, then $[G]$ is 0 and the inequality is trivial.) To take the loop guard G into account, we say in that case that *Inv* is a *sub-martingale* on G.

4.2 Introduction, Informal Explanation and Example of the New Rule

The new rule is presented here, with an informal explanation; just below it we highlight the way in which it differs from the existing rule referred to in Thm. 3.4; then we give an overview of the new rule's proof; and finally we give an informal example. The detailed proof follows in Section §4.3, and fully worked-out examples are given in §5. To distinguish material in this section from the earlier rules above, here we use single-letter identifiers for predicates and expectations.

We say that a function is *antitone* just when $x \leq y \Rightarrow f(x) \geq f(y)$ for all x, y.

Theorem 4.1 (New Variant Rule for Loops). *Let* $I, G \subseteq \Sigma$ *be predicates; let* $V\colon \Sigma \to \mathbb{R}_{\geq 0}$ *be a non-negative real-valued function* not necessarily bounded*; let* p *(for "probability") be a fixed function of type* $\mathbb{R}_{\geq 0} \to (0,1]$*; let* d *(for "decrease") be a fixed function of type* $\mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$*, both of them antitone on strictly positive arguments; and let Com be a* pGCL *program.*

Suppose the following four conditions hold:

(i) I *is a standard invariant of* $\mathtt{while}\,(G)\,\{Com\}$ *, and*

(ii) $G \wedge I \Rightarrow V > 0$ *, and*

(iii) For any $R \in \mathbb{R}_{>0}$ *we have* $p(R)\cdot[G \wedge I \wedge V{=}R] \;\leq\; \mathtt{wp.}Com\mathtt{.}[V \leq R - d(R)]$ *, and*

(iv) V satisfies the "super-martingale" condition that

$$\textit{for any constant } H \textit{ in } \mathbb{R}_{>0} \textit{ we have} \qquad [G \wedge I] \cdot (H \ominus V) \leq \mathtt{wp}.\mathit{Com}.(H \ominus V)\,,$$

where $H \ominus V$ is defined as $\max\{H-V,\, 0\}$.

Then we have $[I] \leq \mathtt{wp}.\mathtt{while}\,(G)\,\{\mathit{Com}\}.1$.

Note that our theorem is stated (and will be proved) in terms of $H \ominus V$. Our justification however for calling *(iv)* a "super-martingale condition" on V is that decrease (in expectation) of V is equivalent to increase of $H \ominus V$. (App. B gives more detail.) Further, in our coming appeal to Thm. 3.5 the expectation *Sub* must be bounded — and V is not (necessarily). Thus we use $H \ominus V$ for arbitrary H instead, each instance of which is bounded by H; and V decreases when $H \ominus V$ increases.

The other reason for using the "inverted" formulation is that pGCL interprets demonic choice by *minimising* over possible final distributions, and so the direction of the inequality in Thm. 3.5 means we must express the "super-martingale property" of V in this complementary way.

As in Thm. 3.4(iii), we have written in the Hoare style $V{=}R$ in the pre-expectation at (iii) above to make V's initial value available (as the real R) in the post-expectation. The overall effect is

> If a predicate I is a standard invariant, and there is a non-negative real-valued variant function V on the state, a super-martingale on $I \wedge G$ with the progress condition that every iteration *Com* of the loop decreases it by at least $d()$ of its initial value with probability at least $p()$ of its initial value, then the loop $\mathtt{while}\,(G)\,\{\mathit{Com}\}$ terminates *AS* from any inital state satisfying I.

The differences from the earlier variant rule Thm. 3.4 are these:

(1) The variant V is now real-valued, with no upper bound (but is bounded below by zero). We call V a *quasi-variant* to distinguish it from traditional integer-valued variants.
(2) Quasi-variants are *not* required to decrease by a fixed non-zero amount with a fixed non-zero probability. Instead there are two functions p, d that give for each variant-value how much *Com* must decrease it (at least) and with what probability (at least). The only restriction on those functions (aside from the obvious ones) is that they be antitone, i.e. that for larger arguments they must give equal-or-smaller (but never zero) values. The reason for requiring p and d to be antitone is to exclude Zeno-like behavior where the variant decreases less and less, and/or with less and less probability. Otherwise, each loop iteration could decrease the variant by a positive amount with positive probability –bringing it ever closer to zero– but never actually reaching the zero that implies negation of the guard, and thus termination.
(3) Quasi-variants *are* required to be super-martingales: that from every state satisfying $G \wedge I$ the expected value of the quasi-variant after *Com* cannot increase.
Note that Thm. 3.4 did not have a super-martingale assumption: although the probability that *VInt* decreased by at least 1 was required there to be at least ε, the change in expected value of *VInt* was unconstrained. For example, if with the remaining probability $1{-}\varepsilon$ it increased by a lot (but still not above *High*), then its expected value could actually increase as well.

A simple example of the power of Thm. 4.1 (Theme A in §1) is in fact the symmetric random walk mentioned earlier. Let the state-space be the integers x, and let each loop iteration when $x{\neq}0$ either decrease x by 1 or increase it by 1 with equal probability. *AST* is out of reach of the earlier rule Thm. 3.4 because x is not bounded above, and out of reach of some others' rules too, because the expected time to termination is infinite [Ferrer Fioriti and Hermanns 2015]. Yet termination at $x{=}0$ is shown immediately with Thm. 4.1 by taking $V{=}|x|$, trivially an exact martingale when $x{\neq}0$, and $p{=}1/2$ and $d{=}1$.

4.3 Rigorous Proof of Thm. 4.1

We begin with an informal description of the strategy of the proof that follows.

A. We choose an arbitrary real value $H{>}0$ and temporarily strengthen the loop's guard by conjoining $V{\leq}H$. From the antitone properties of p, d we know that each execution of *Com* with that strengthened guard decreases quasi-variant V by at least $d(H)$ with probability at least $p(H)$. Using that to "discretise" V, making it an integer bounded above and below, we can appeal to the earlier Thm. 3.4 to show that this guard-strengthened loop terminates *AS* for any H.
B. Using the super-martingale property of V, we argue that the probability of "bad" escape to $V{>}H$ decreases to zero as H increases: for escape from the strengthened loop to $V{>}H$ with some probability e say implies a contribution of at least $e \cdot H$ to V's expected value at that point. But that expected value cannot exceed V's original value, because V is a super-martingale. (For this we appeal to Thm. 3.5 after converting V into a sub-martingale as required there.) Thus as H gets larger e must get smaller.
C. Since e approaches 0 as H increases indefinitely, we argue finally that, wherever we start, we can make the probability of escape to $V{>}H$ as small as we like by increasing H sufficiently; complementarily we are making the only remaining escape probability, i.e. of "good" escape to $\neg G$, as close to 1 as we like. Thus it equals 1, since H was arbitrary. Because this last argument depends essentially on increasing H without bound, it means that p, d must be defined, non-zero and antitone on *all* positive reals, not only on those resulting from $V(\sigma)$ on some state σ the program happens to reach. This is particularly important when V is bounded. (See §8.2.)

We now give the rigorous proof of Thm. 4.1, following the strategy explained just above.

Proof. (of Thm. 4.1)
Let V be a quasi-variant for $\texttt{while}\,(G)\,\{Com\}$, satisfying p, d progress for some p, d as defined in the statement of the theorem, and recall that I is a standard invariant for that loop.

A. ——— *For any H, the loop (6) below terminates AS from any initial state satisfying I.*
Fix arbitrary H in $\mathbb{R}_{>0}$, and strengthen the loop guard G of $\texttt{while}\,(G)\,\{Com\}$ with the conjunct $V{\leq}H$. We show that

$$[I] \quad \leq \quad \texttt{wp.while}\,(G \wedge V{\leq}H)\,\{Com\}\texttt{.}\mathbf{1}\,, \tag{6}$$

i.e. that standard invariant I describes a set of states from which the loop (6) terminates *AS*.

We apply Thm. 3.4 to (6), after using ceiling $\lceil - \rceil$ to make an integer-valued variant *VInt*, and with other instantiations as follows:

$$\begin{array}{c} \mathit{Inv} := I \qquad \mathit{Guard} := G \wedge V{\leq}H \\ \mathit{VInt} := \left\lceil \frac{V}{d(H)} \right\rceil \qquad \mathit{Low} := 0 \qquad \mathit{High} := \left\lceil \frac{H}{d(H)} \right\rceil \qquad \varepsilon := p(H) \end{array} \tag{7}$$

The *VInt* can be thought of as a *discretised* version of V — the original V moves between 0 and H with down-steps of at least $d(H)$ while integer *VInt* moves between 0 and *High* with down-steps of at least 1. In both cases, the down-steps occur with probability at least $p(H)$.

We now verify that our choices (7) satisfy the assumptions of Thm. 3.4:

(1) *Inv* is a standard invariant of (6) because I is by assumption a standard invariant of the loop $\texttt{while}\,(G)\,\{Com\}$, and the only difference is that (6) has a stronger guard.

(2) Now note that $V \leq H$ implies $\lceil V/a \rceil \leq \lceil H/a \rceil$ for any strictly positive a. Then

$$\begin{array}{lll} & \mathit{Guard} \wedge \mathit{Inv} & \\ \iff & (G \wedge V \leq H) \wedge I & \text{instantiations } \mathit{Guard}, \mathit{Inv} \\ \implies & 0 < V \leq H & G \wedge I \Rightarrow 0 < V \text{ assumed at Thm. 4.1 (ii)} \\ \implies & 0 < \lceil V/d(H) \rceil \leq \lceil H/d(H) \rceil & \text{remark above and } d(H) > 0 \\ \implies & \mathit{Low} < \mathit{VInt} \leq \mathit{High}\,. & \text{instantiations } \mathit{Low}, \mathit{VInt}, \mathit{High} \end{array}$$

(3) In this final section of Step (A) we will write in an explicit style that relies less on Hoare-logic conventions and more on exposing clearly the types involved and the role of the initial- and final state. In this style, our assumption for appealing to Thm. 3.4 is that for all (initial) states σ we have

$$\begin{array}{rl} & p(H) \cdot [G(\sigma) \wedge V(\sigma) \leq H \wedge I(\sigma)] \qquad (8) \\ \leq & \mathsf{wp}\,.\,\mathit{Com}\,.\,(\lambda\sigma'.\,[\mathit{VInt}(\sigma') < \mathit{VInt}(\sigma)])(\sigma)\,. \qquad (9) \end{array}$$

Here both the *lhs* and *rhs* are real-valued expressions in which an arbitrary initial state σ appears free. On the left G, I are predicates on Σ, and V is a non-negative real-valued function on Σ, and p, H are constants of type $\mathbb{R}_{>0} \to \mathbb{R}_{>0}$ and $\mathbb{R}_{>0}$ respectively.
On the right $\mathsf{wp}\,.\,\mathit{Com}\,.\,(-)$ is a (weakest pre-) expectation, a real-valued function on Σ; applying it to the initial state –the final (σ) in (9) at *rhs*– produces a non-negative real scalar. The second argument $(-)$ of $\mathsf{wp}\,.\,\mathit{Com}\,.\,(-)$ is a post-expectation, again a function of type $\Sigma \to \mathbb{R}_{\geq 0}$, but $\mathsf{wp}\,.\,\mathit{Com}$ takes that $(-)$'s expected value over the *final* distribution(s) that *Com* reaches from σ — for mnemonic advantage, we bind its states with σ'. And using σ' also allows us to refer in $(-)$ to the initial state as σ, not captured by $(\lambda\sigma'.\cdots)$, so that we can compare the initial $\mathit{VInt}(\sigma)$ and final $\mathit{VInt}(\sigma')$ values of *VInt* as required.
What we have now is our assumption of progress for the original loop $\texttt{while}\,(G)\,\{\mathit{Com}\}$, which was

$$\begin{array}{rl} & p(V(\sigma)) \cdot [G(\sigma) \wedge I(\sigma)] \\ \leq & \mathsf{wp}\,.\,\mathit{Com}\,.\,(\lambda\sigma'.\,\big[V(\sigma') \leq V(\sigma) - d\big(V(\sigma)\big)\big])(\sigma)\,, \end{array} \qquad (10)$$

and we must use (10), together with the antitone properties of p, d to show (8)≤(9). We begin with (8) and reason

$$\begin{array}{lll} & p(H) \cdot [G(\sigma) \wedge V(\sigma) \leq H \wedge I(\sigma)] & (8) \text{ above} \\ = & p(H) \cdot [G(\sigma) \wedge 0 < V(\sigma) \leq H \wedge I(\sigma)] & G \wedge I \Rightarrow V > 0 \text{ by assumption Thm. 4.1(ii)} \\ \leq & p(V(\sigma)) \cdot [G(\sigma) \wedge 0 < V(\sigma) \leq H \wedge I(\sigma)] & V(\sigma) \leq H;\ p \text{ antitone and defined on } V(\sigma)\ [9] \\ \leq & p(V(\sigma)) \cdot [G(\sigma) \wedge I(\sigma)] & \text{drop conjunct: } [A \wedge B \wedge C] \leq [A \wedge C] \\ \leq & \mathsf{wp}\,.\,\mathit{Com}\,.\,(\lambda\sigma'.\,\big[V(\sigma') \leq V(\sigma) - d\big(V(\sigma)\big)\big])(\sigma)\,. & \text{assumption (10) above} \end{array}$$

[9] Here potentially the value of $p(0)$ is used on the left, when $V(\sigma)$ is zero; but because $[\cdots 0 < V(\sigma) \cdots] = 0$ in that case, it makes no different what $p(0)$'s value is. The antitone property applies only for positive arguments.

Now continuing only within the $[-]$ of the post-expectation we have [10]

$$\begin{array}{lll}
& V(\sigma') \le V(\sigma) - d\big(V(\sigma)\big) & \\
\Longrightarrow & \Big\lceil V(\sigma')/d(H) \Big\rceil \le \Big\lceil V(\sigma)/d(H) - d\big(V(\sigma)\big)/d(H) \Big\rceil & d(H){>}0,\ \lceil - \rceil \text{ monotonic} \\
\Longrightarrow & \Big\lceil V(\sigma')/d(H) \Big\rceil \le \Big\lceil V(\sigma)/d(H) \Big\rceil - 1 & V(\sigma) \le H,\ d \text{ antitone, } \mathit{lhs}\ (8) \\
\Longrightarrow & \Big\lceil V(\sigma')/d(H) \Big\rceil < \Big\lceil V(\sigma)/d(H) \Big\rceil & \\
\Longrightarrow & \mathit{VInt}(\sigma') < \mathit{VInt}(\sigma)\ . & \text{definition } \mathit{VInt}
\end{array}$$

Placing the last line back within $\mathtt{wp}\,.\,\mathit{Com}\,.\,(\lambda\sigma'.\,[-])(\sigma)$ gives what was required at (9) and establishes (6) — that escape from $0{<}V{\le}H$ occurs *AS* from any initial state satisfying I.

B. ___ *Loop (6)'s probability of termination at $\neg G$ tends to 1 as $H{\to}\infty$.*
For the probabilistic invariant, i.e. sub-martingale *Sub* in Theorem 3.5, we choose $H{\ominus}V$. Note that, as required by Thm. 3.5, expectation *Sub* is bounded (by H). Let predicate *Term* be I which from (6) we know ensures *AST* of the modified loop. Thus the assumptions of Thm. 3.5 are satisfied: reasoning from its conclusion we have

$$\begin{array}{lll}
& [I]\cdot H{\ominus}V \ \le\ \mathtt{wp.while}\,(G \wedge V{\le}H)\,\{\mathit{Com}\}\,.\,([\neg(G \wedge V{\le}H)]\cdot H{\ominus}V) & \\
\Longleftrightarrow & [I]\cdot H{\ominus}V \ \le\ \mathtt{wp.while}\,(G \wedge V{\le}H)\,\{\mathit{Com}\}\,.\,([\neg G]\cdot H{\ominus}V) & V{>}H \Rightarrow H{\ominus}V{=}0 \\
\Longleftrightarrow & [I]\cdot 1{\ominus}V/H \ \le\ \mathtt{wp.while}\,(G \wedge V{\le}H)\,\{\mathit{Com}\}\,.\,([\neg G]\cdot 1{\ominus}V/H) & \text{scaling (4) by } 1/H \\
\Longrightarrow & 1{\ominus}V/H\cdot [I] \ \le\ \mathtt{wp.while}\,(G \wedge V{\le}H)\,\{\mathit{Com}\}\,.\,[\neg G]\ , & \text{monotonicity}
\end{array}$$

that is, recalling (3), that from any initial state satisfying I the loop (6) terminates in a state satisfying $\neg G$ with probability at least $1{\ominus}V/H$. As required, that probability (for fixed initial state) tends to 1 as H tends to infinity.

C. ___ *The original loop terminates AS from any initial state satisfying I.*
From App. A, instantiating $A{:=}\,G$ and $B{:=}\,V{\le}H$, we have for any H that

$$\mathtt{wp.while}\,(G \wedge V{\le}H)\,\{\mathit{Com}\}\,.\,[\neg G] \qquad \le \qquad \mathtt{wp.while}\,(G)\,\{\mathit{Com}\}\,.\,[\neg G]$$

and, referring to the last line in (B) just above, we conclude $(1{\ominus}V/H){\cdot}[I] \le \mathtt{wp.while}\,(G)\,\{\mathit{Com}\}\,.\,[\neg G]$. Since that holds for any H no matter how large, we have finally that

$$[I] \qquad \le \qquad \mathtt{wp.while}\,(G)\,\{\mathit{Com}\}\,.\,[\neg G] \qquad \le \qquad \mathtt{wp.while}\,(G)\,\{\mathit{Com}\}\,.\,\mathbf{1}\ ,$$

that is that from any initial state satisfying I the loop $\mathtt{while}\,(G)\,\{\mathit{Com}\}$ terminates *AS*. □

5 CASE STUDIES

In this section, we examine a few (mostly) non-trivial examples to show the effectiveness of Thm. 4.1. For all examples we provide a p, d quasi-variant V that proves *AST*; and we will always choose p, d so that they are strictly positive and antitone. We will not provide proofs of the p, d properties, because they will be self-evident and are in any case "external" mathematical facts. We do however carefully set-out any proofs that depend on the program text: that $V{=}0$ indicates termination, that V satisfies the super-martingale property, and that p, d, and V satisfy the progress condition.

For convenience in these examples, we define a derived expectation transformer awp, over terminating straight-line programs only (as our loop bodies are, in this section), that "factors out"

[10] This reduces clutter, and in general $A{\Rightarrow}B$ implies $[A] \le [B]$, and $\mathtt{wp}\,.\,\mathit{Com}\,.\,(-)$ is itself monotonic for any *Com*.

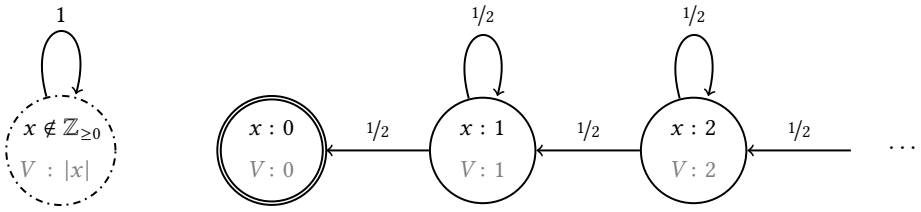

Fig. 1. Execution of the negative binomial loop. The solid nodes represent program states and moreover the doubly-circled node represents a state in which the loop has terminated. The leftmost dash-dotted node represents the *collection* of all states in which the value of x is not a non-negative integer (from where the random walk will indeed not terminate). Inside the nodes we give the variable valuations as well as the values of the variant $V = |x|$ in each state. The values of p and d are constantly $1/2$ and 1, respectively.

the $(H\ominus)$; it has the same definition as of wp in Table 1 except that nondeterminism is interpreted angelically rather than demonically: that is, we define

$$\mathsf{awp}\,.\,\{C_1\}\,\Box\,\{C_2\}\,.\,f \qquad = \qquad \max\{\mathsf{awp}\,.\,C_1\,.\,f,\ \mathsf{awp}\,.\,C_2\,.\,f\}\ ,$$

and otherwise as for wp (except for loops, which we do not need here). A straightforward structural induction then shows that for straight-line programs Com, constant H and any expectation V that

$$H \ominus \mathsf{awp}\,.\,Com\,.\,V \qquad \leq \qquad \mathsf{wp}\,.\,Com\,.\,(H\ominus V)\ . \tag{11}$$

And from there we have immediately that

$$V \geq \mathsf{awp}\,.\,Com\,.\,V \qquad \Longrightarrow \qquad H\ominus V \leq \mathsf{wp}\,.\,Com\,.\,(H\ominus V)\ , \tag{12}$$

and finally therefore that

$$V \geq [G \wedge I]\cdot \mathsf{awp}\,.\,Com\,.\,V \qquad \Longrightarrow \qquad [G \wedge I]\cdot(H\ominus V) \leq \mathsf{wp}\,.\,Com\,.\,(H\ominus V)\ , \tag{13}$$

since if $G \wedge I$ holds then (13) reduces to (12) and, if it does not hold, both sides of (13) are trivially true. Thus when the loop body is a straight-line program, by establishing *lhs* (13) we establish also *rhs* (13) as required by Thm. 4.1(iv). We stress that awp is used here for concision and intuition only: applied only to finite, non-looping programs, it can always be replaced by wp.

Thus *lhs* (13) expresses clearly and directly that V is a super-martingale when $G \wedge I$ holds, and handles any nondeterminism correctly in that respect: because awp *maximises* rather than minimises over nondeterministic outcomes (the opposite of wp), the super-martingale inequality $(\geq)$ holds for every individual outcome, as required.

In §8.3 we discuss the reasons for not using awp in Thm. 4.1 directly, i.e. not eliminating "$H\ominus$" at the very start: in short, it is because our principal reference [McIver and Morgan 2005] does not support awp.

5.1 The Negative-Binomial Loop

Our first example is also proved by other *AST* rules, so we do not need the extra power of Thm. 4.1 for it; but we begin with this to illustrate Theme B with a familar example how Thm. 4.1 is used in formal reasoning over program texts.

Description of the loop. Consider the following while loop over the real-valued variable x:

$$\texttt{while}\,(x{\neq}0)\,\{\ x \coloneqq x{-}1\ {}_{1/2}\!\oplus\ \texttt{skip}\ \}\ . \tag{14}$$

An interpretation of this loop as a transition system is illustrated in Figure 1.

Intuitively, this loop keeps flipping a coin until it flips, say, heads x times (not necessarily in a row); every time it flips tails, the loop continues without changing the program state.

We call it the negative binomial loop because its runtime is distributed according to a negative binomial distribution (with parameters x and $1/2$), and thus the expected runtime is linear (on average $2x$ loop iterations) even though it allows for infinite executions, namely those runs of the program that flip heads fewer than x times and then keep flipping tails ad infinitum.

A subtle intricacy is that this loop will not terminate at all, if x is initially not a *non-negative integer*, because then the execution of the loop never reaches a state in which $x{=}0$. This is where we use Theorem 4.1's ability of incorporating an invariant into the *AST* proof, as standard arguments over loop termination do.

Proof of almost-sure termination. The guard is given by $G \;=\; x{\neq}0\,,$
and the loop body by $Com \;=\; \{x \coloneqq x-1\}\ {}_{1/2}\!\oplus \{\mathtt{skip}\}\,.$
And with the standard invariant $I \;=\; x{\in}\mathbb{Z}_{\geq 0}\,,$
we can now prove *AST* of the loop with an appropriate p, d and quasi-variant V:

$$V \;=\; |x|, \qquad \text{for } d \;=\; 1 \quad \text{and} \quad p \;=\; 1/2\,.$$

Notice that d, p are strictly speaking constant functions mapping any positive real v to $1, 1/2$ respectively. Intuitively, this choice of I, V, p, and d tells us that if x is a positive integer different from 0, then after one iteration of the loop body (a) x is still a non-negative integer (by invariance of I) and (b) the distance of x from 0 has decreased by at least 1 with probability at least $1/2$ (implied by the progress condition).

We first check that $I = x{\in}\mathbb{Z}_{\geq 0}$ is indeed an invariant:

$$\begin{aligned}
[G]\cdot[I] \;=\; [x\neq 0]\cdot[x\in\mathbb{Z}_{\geq 0}] \;&=\; [x\in\mathbb{Z}_{>0}] \\
&\leq\; \frac{1}{2}\Big([x\in\mathbb{Z}_{>0}] + [x\in\mathbb{Z}_{\geq 0}]\Big) \\
&=\; \frac{1}{2}\Big([x{-}1\in\mathbb{Z}_{\geq 0}] + [x\in\mathbb{Z}_{\geq 0}]\Big) \\
&=\; \mathtt{wp}\,.\,\{x\coloneqq x-1\}\ {}_{1/2}\!\oplus\{\mathtt{skip}\}\,.\,[x\in\mathbb{Z}_{\geq 0}] \\
&=\; \mathtt{wp}\,.\,Com\,.\,[I]\,.
\end{aligned}$$

Next, the second precondition of Theorem 4.1 is satisfied because of

$$G\wedge I \iff x{\neq}0\wedge x{\in}\mathbb{Z}_{\geq 0} \implies x{\neq}0 \implies |x|{>}0 \iff V{>}0\,.$$

Furthermore, V satisfies the super-martingale property:

$$\begin{aligned}
[G\wedge I]\cdot \mathtt{awp}\,.\,Com\,.\,V \;&=\; [x{\neq}0\wedge x{\in}\mathbb{Z}_{\geq 0}]\cdot \mathtt{awp}\,.\,(\{x\coloneqq x-1\}\ {}_{1/2}\!\oplus\{\mathtt{skip}\})\,.\,|x| \\
&=\; [x\in\mathbb{Z}_{>0}]\cdot\frac{1}{2}\cdot\Big(|x-1|+|x|\Big) \\
&=\; [x\in\mathbb{Z}_{>0}]\cdot\Big(|x|-\frac{1}{2}\Big) \\
&\leq\; [x\in\mathbb{Z}_{>0}]\cdot|x| \\
&\leq\; |x| \\
&=\; V\,.
\end{aligned}$$

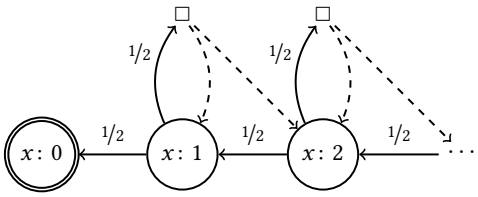

Fig. 2. Execution of the demonically fair random walk. The □ nodes together with the dashed arrows represent demonic choices. The value of the variant is equal to the value of x in each state. The values of p and d are constantly $1/2$ and 1, respectively.

Lastly, V, p, and d satisfy the progress condition for all R:

$$
\begin{aligned}
& p(R) \cdot [G \wedge I \wedge V{=}R] \;\leq\; \mathtt{wp}.\mathit{Com}.[V \leq R - d(R)] \\
\iff\; & \frac{1}{2} \cdot [x{\neq}0 \wedge x{\in}\mathbb{Z}_{\geq 0} \wedge |x|{=}R] \;\leq\; \mathtt{wp}.\{x \coloneqq x-1\} \,{}_{1/2}\!\oplus\, \{\mathtt{skip}\}.[|x| \leq R{-}1] \\
\iff\; & \frac{1}{2} \cdot [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; \mathtt{wp}.\{x \coloneqq x-1\} \,{}_{1/2}\!\oplus\, \{\mathtt{skip}\}.[|x| \leq R{-}1] \\
\iff\; & \frac{1}{2} \cdot [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; \frac{1}{2} \cdot \Big([|x{-}1| \leq R{-}1] + [|x| \leq R{-}1] \Big) \\
\iff\; & [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; \Big([|x{-}1| \leq R{-}1] + [|x| \leq R{-}1] \Big) \\
\iff\; & [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \cdot \Big([|x{-}1| \leq R{-}1] + [|x| \leq R{-}1] \Big) \\
\iff\; & [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \cdot (1+0) \\
\iff\; & [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \;\leq\; [x{\in}\mathbb{Z}_{>0} \wedge |x|{=}R] \\
\iff\; & \mathsf{true}\ .
\end{aligned}
$$

This shows that all preconditions of Theorem 4.1 are satisfied: thus we have $[x{\in}\mathbb{Z}_{\geq 0}] \leq \mathtt{wp}.(14).\mathbf{1}$, i.e. that the negative binomial loop terminates almost-surely from all initial states in which x is a non-negative integer.

5.2 The Demonically Fair Random Walk

Next, we consider a while loop that contains both probabilistic- and demonic choice.

Description of the loop. Consider the following while loop:

$$
\begin{array}{l}
\mathtt{while}\,(x > 0)\,\{ \\
\qquad \big\{x \coloneqq x-1\big\} \,{}_{1/2}\!\oplus\, \big\{\{x \coloneqq x+1\} \,\square\, \{\mathtt{skip}\}\big\} \\
\}
\end{array}
$$

In order not to clutter the reasoning below, we assume without loss of generality that for this example x is of type $\mathbb{N}$. The execution of the loop is illustrated in Figure 2.

The motivation for this loop is the recursive procedure P inspired by an example of Olmedo et al. [2016]; its definition is

$$
P \;\rhd\; \big\{\mathtt{skip}\big\} \,{}_{1/2}\!\oplus\, \big\{\, \mathtt{call}\ P;\ \{\mathtt{call}\ P\} \,\square\, \{\mathtt{skip}\} \big\}\ ,
$$

and we have rewritten it as a loop by viewing it as a random walk of a particle x whose position represents the height of the call stack. Intuitively, the loop keeps moving x in a random and demonic fashion until the particle hits the origin 0 (empty call stack, all procedure calls have terminated). For

that at each stage it either with probability $1/2$ decrements the position of x by one (procedure call terminates after `skip`; call stack decremented by one), or with probability $1/2$ it performs a demonic choice between incrementing the position of x by one (perform two consecutive procedure calls, then terminate; call stack in effect incremented by one ($+\,2-1=\,+1$)) or letting x remain at its position (perform one procedure call, then terminate; call stack in effect unchanged ($+\,1-1=0$)).

Proof of almost-sure termination. The loop guard is given by $G = x{>}0$ and the loop body by

$$Com \;=\; \Big\{x \coloneqq x-1\Big\} \;{}_{1/2}\!\oplus\; \Big\{\{x \coloneqq x+1\} \,\square\, \{\mathtt{skip}\}\Big\}\;.$$

We now prove *AST* of the loop by choosing the standard invariant $I = \mathsf{true}$ [11] and an appropriate p, d and quasi-variant V:

$$V \;=\; x, \qquad \text{for } d \;=\; 1 \quad \text{and} \quad p \;=\; 1/2\,.$$

Intuitively this choice of V, p, and d tells us that the value of x decreases with probability at least $1/2$ by at least 1 through an iteration of the loop body in the case that initially $x{>}0$.

The second precondition of Theorem 4.1 is satisfied because $G \wedge I \iff x{>}0 \iff V{>}0$. Furthermore, V satisfies the super-martingale property:

$$\begin{aligned}
[G \wedge I] \cdot \mathsf{awp}\,.\,Com\,.\,V \;&=\; [x{>}0] \cdot \mathsf{awp}\,.\,\{x \coloneqq x-1\} \;{}_{1/2}\!\oplus\; \{\{x \coloneqq x+1\} \,\square\, \{\mathtt{skip}\}\}\,.\,x \\
&=\; [x{>}0] \cdot \frac{1}{2} \cdot (x - 1 + \max\{x+1,\, x\}) \\
&=\; [x{>}0] \cdot \frac{1}{2} \cdot (x - 1 + x + 1) \\
&=\; [x{>}0] \cdot x \\
&\leq\; x \\
&=\; V\,.
\end{aligned}$$

Lastly, V, p, and d satisfy the progress condition for all R:

$$\begin{aligned}
& p(R) \cdot [G \wedge I \wedge V{=}R] \;\leq\; \mathsf{wp}\,.\,Com\,.\,[V \leq R - d(R)] \\
\iff\; & \frac{1}{2} \cdot [x{>}0 \wedge \mathit{true} \wedge x{=}R] \;\leq\; \mathsf{wp}\,.\,\Big\{x \coloneqq x-1\Big\} \;{}_{1/2}\!\oplus\; \Big\{\{x \coloneqq x+1\} \,\square\, \{\mathtt{skip}\}\Big\}\,.\,[x \leq R{-}1] \\
\iff\; & \frac{1}{2} \cdot [x{>}0 \wedge x{=}R] \;\leq\; \frac{1}{2} \cdot \Big([x{-}1 \leq R{-}1] + \max\{[x{+}1 \leq R{-}1],\, [x \leq R{-}1]\} \Big) \\
\iff\; & [x{>}0 \wedge x{=}R] \;\leq\; [x \leq R] + [x \leq R{-}1] \\
\iff\; & [x{>}0 \wedge x{=}R] \;\leq\; [x \leq x] + [x \leq x{-}1] \\
\iff\; & [x{>}0 \wedge x{=}R] \;\leq\; 1 + 0 \\
\iff\; & \mathsf{true}\,.
\end{aligned}$$

This shows that all preconditions of Theorem 4.1 are satisfied and as a consequence the demonic random walk loop above terminates almost-surely. Interestingly, the procedure P' given by

$$P' \;\rhd\; \{\mathtt{skip}\} \;{}_{1/2}\!\oplus\; \{\mathtt{call}\; P';\; \mathtt{call}\; P';\; \{\mathtt{call}\; P'\} \,\square\, \{\mathtt{skip}\}\}\;,$$

i.e. potentially three consecutive procedure calls instead of two [Olmedo et al. 2016], is not *AST*: it terminates with probability only $(\sqrt{5}-1)/2 < 1$.

[11] Predicate true is an invariant for any loop whose body is terminating, e.g. is itself loop-free.

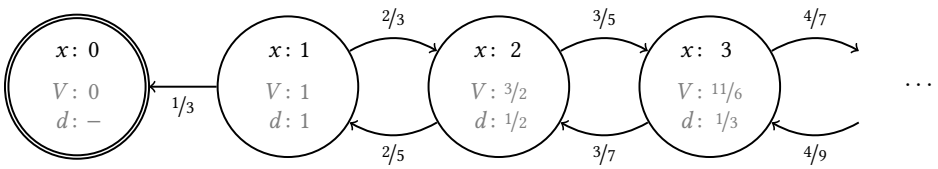

Fig. 3. Execution of the fair-in-the-limit random walk. Inside the nodes we give the valuations of variable x as well as the values of the variant V and the decrease function d. The value of p is constantly $1/3$. Note that in Thm. 4.1 it does not matter what d's value is when $V{=}0$, because the *lhs* of (iii) is zero in that case.

5.3 The Fair-in-the-Limit Random Walk

While so far we have considered constant probabilities and constant decreases, we now consider a while loop requiring use of a non-constant decrease function d.

Description of the loop. Consider the following while loop:

$$\begin{array}{l}\texttt{while}\,(x > 0)\,\{\\ \quad q \coloneqq {}^{x}\!/_{2x+1};\\ \quad \{x \coloneqq x-1\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\}\\ \}\end{array}$$

Assume again that $x{\in}\mathbb{N}$. The execution of the loop is illustrated in Figure 3.

Intuitively, the loop models an asymmetric random walk of a particle x, terminating when the particle hits the origin 0. In one iteration of the loop body, the program either with probability ${}^{x}\!/_{2x+1}$ decrements the position of x by one, or with probability ${}^{x+1}\!/_{2x+1}$ increments the position of x by one. The further the particle x is away from 0, the more fair becomes the random walk since ${}^{x}\!/_{2x+1}$ approaches $1/2$ asymptotically. Yet, it is not so obvious that this random walk indeed also terminates with probability 1.

Proof of almost-sure termination. The loop guard is given by $G = x{>}0$ and the loop body by

$$Com \;=\; q \coloneqq {}^{x}\!/_{2x+1};\ \{x \coloneqq x-1\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\}\,.$$

We now prove almost-sure termination of the loop by choosing standard invariant $I = \mathsf{true}$ and an appropriate p, d quasi-variant V:

$$V \;=\; H_x, \qquad \text{for } d(v) \;=\; \begin{cases}\frac{1}{x}, & \text{if } v > 0 \text{ and } v \in (H_{x-1},\, H_x]\\ 1, & \text{if } v = 0\end{cases} \qquad \text{and} \quad p(v) \;=\; \frac{1}{3}\,,$$

where H_x is the x-th harmonic number.[12] Notice that the variant V is non-affine here, i.e. not of the form $a + bx + cq$, and we will show below that no affine variant can satisfy a super-martingale property. Intuitively our choice of p and d tells us that the variant V, i.e. the harmonic number of the value of x, decreases with probability at least $1/3$ by at least $\frac{1}{x}$ through an iteration of the loop body in case that initially $x > 0$.

The second precondition of Theorem 4.1 is satisfied because

$$G \wedge I \iff x{>}0 \iff H_x{>}0 \iff V{>}0\,.$$

[12] $H_x = \sum_{n=1}^{x} \frac{1}{n}$. Notice that $H_0 = 0$.

Furthermore, V satisfies the super-martingale property:

$$\begin{aligned}
[G]\cdot \mathsf{awp}\,.\,Com\,.\,V &= [x{>}0]\cdot \mathsf{awp}\,.\,q \coloneqq {}^{x}\!/_{2x+1};\ \{x \coloneqq x-1\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\}\,.\,H_x \\
&= [x{>}0]\cdot \mathsf{awp}\,.\,q \coloneqq {}^{x}\!/_{2x+1}\,.\,(q\cdot H_{x-1} + (1{-}q)\cdot H_{x+1}) \\
&= [x{>}0]\cdot \left(\frac{x}{2x{+}1}\cdot H_{x-1} + \left(1-\frac{x}{2x{+}1}\right)\cdot H_{x+1}\right) \\
&= [x{>}0]\cdot \left(\frac{x}{2x{+}1}\cdot \left(H_x - \frac{1}{x}\right) + \left(\frac{x{+}1}{2x{+}1}\right)\cdot \left(H_x + \frac{1}{x{+}1}\right)\right) \\
&= [x{>}0]\cdot \left(\left(\frac{x}{2x{+}1} + \frac{x{+}1}{2x{+}1}\right)\cdot H_x - \frac{1}{2x{+}1} + \frac{1}{2x{+}1}\right) \\
&= [x{>}0]\cdot H_x \\
&\leq H_x \\
&= V\ .
\end{aligned}$$

Lastly, V, p, and d satisfy the progress condition for all R. Notice that $d(H_x) = {}^{1}\!/_{x}$ and consider the following:

$$\begin{aligned}
& p(R)\cdot [G \wedge I \wedge V{=}R] \ \leq\ \mathsf{wp}\,.\,Com\,.\,[V \leq R - d(R)] \\
\iff\ & \frac{1}{3}\cdot [x{>}0 \wedge H_x{=}R] \ \leq\ \mathsf{wp}\,.\,q \coloneqq {}^{x}\!/_{2x+1};\ \{x \coloneqq x-1\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\}\,.\,[H_x \leq R - d(R)] \\
\iff\ & \frac{1}{3}\cdot [x{>}0 \wedge H_x{=}R] \ \leq\ \mathsf{wp}\,.\,q \coloneqq {}^{x}\!/_{2x+1}\,.\,(q\cdot [H_{x-1} \leq R - d(R)] + (1{-}q)\cdot [H_{x+1} \leq R - d(R)]) \\
\iff\ & \frac{1}{3}\cdot [x{>}0 \wedge H_x{=}R] \ \leq\ \frac{x}{2x{+}1}\cdot [H_{x-1} \leq R - d(R)] + \left(1-\frac{x}{2x{+}1}\right)\cdot [H_{x+1} \leq R - d(R)] \\
\iff\ & \frac{1}{3}\cdot [x{>}0 \wedge H_x{=}R] \ \leq\ \frac{x}{2x{+}1}\cdot [H_{x-1} \leq R - d(R)] + \left(\frac{x{+}1}{2x{+}1}\right)\cdot [H_{x+1} \leq R - d(R)] \\
\iff\ & \frac{1}{3}\cdot [x{>}0 \wedge H_x{=}R] \ \leq\ \frac{x}{2x{+}1}\cdot \left[H_{x-1} \leq H_x - \frac{1}{x}\right] + \left(\frac{x{+}1}{2x{+}1}\right)\cdot \left[H_{x+1} \leq H_x - \frac{1}{x}\right] \\
\iff\ & [x{>}0]\cdot \frac{1}{3} \ \leq\ \left(\frac{x}{2x{+}1}\cdot 1 + \frac{x{+}1}{2x{+}1}\cdot 0\right) \\
\iff\ & [x{>}0]\cdot \frac{1}{3} \ \leq\ \frac{x}{2x{+}1} \\
\iff\ & \mathsf{true}\ .
\end{aligned}$$

This shows that all preconditions of Theorem 4.1 are satisfied and as a consequence the fair-in-the-limit random walk terminates almost-surely.

Proof of non-existence of an affine variant. For this program, there exists *no affine variant* that satisfies the super-martingale property as used e.g. by Chatterjee et al. [2017]. Any affine [13] variant V would have to be of the form

$$V \ =\ a + bx + cq\,,$$

[13] Some authors call this a *linear* variant.

for some (positive) coefficients a, b, c. [14] Now we attempt to check the super-martingale property for a variant of that form:

$$
\begin{aligned}
& [G] \cdot \mathsf{awp}\,.\,Com\,.\,V \\
=\ & [x{>}0] \cdot \mathsf{awp}\,.\,q \coloneqq {}^{x}\!/_{2x+1};\ \{x \coloneqq x-1\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\}\,.\,(a + bx + cq) \\
=\ & [x{>}0] \cdot \mathsf{awp}\,.\,q \coloneqq {}^{x}\!/_{2x+1}\,.\,(q \cdot (a + b(x{-}1) + cq) + (1{-}q) \cdot (a + b(x{+}1) + cq)) \\
=\ & [x{>}0] \cdot \mathsf{awp}\,.\,q \coloneqq {}^{x}\!/_{2x+1}\,.\,(a - 2bq + bx + b + cq) \\
=\ & [x{>}0] \cdot \left(a - 2b \cdot \frac{x}{2x+1} + bx + b + c \cdot \frac{x}{2x+1}\right) \\
\overset{!}{\leq}\ & a + bx + cq \\
=\ & V\ .
\end{aligned}
$$

If $x \leq 0$ this is trivially satisfied. If $x{>}0$, then the above is satisfied iff

$$
\begin{aligned}
& a - 2b \cdot \frac{x}{2x+1} + bx + b + c \cdot \frac{x}{2x+1} \ \leq\ a + bx + cq \\
\iff\quad & -2b \cdot \frac{x}{2x+1} + b + c \cdot \frac{x}{2x+1} \ \leq\ cq\ ,
\end{aligned}
$$

which is only satisfiable for all possible valuations of q and $x{>}0$ iff $b = c = 0$. Thus if V is forced to be affine, then V has to be constantly a, for $a \geq 0$. Indeed, a is a super-martingale. However, it is clear that a constant V cannot possibly indicate termination as

$$
[V = 0] \ =\ 1 \ \neq\ [x \leq 0] \ =\ [\neg G]\ .
$$

Thus, there cannot exist an affine variant that satisfies the super-martingale property.

5.4 The Escaping Spline

We now consider a while loop where we we will make use of both non-constant probability function p and non-constant decrease function d.

Description of the loop. Consider the following while loop:

$$
\begin{aligned}
&\texttt{while}\,(x > 0)\,\{ \\
&\qquad q \coloneqq {}^{1}\!/_{x+1}; \\
&\qquad \{x \coloneqq 0\}\ {}_{q}\!\oplus\ \{x \coloneqq x+1\} \\
&\}
\end{aligned}
$$

Assume again that $x{\in}\mathbb{N}$. The execution of the loop is illustrated in Figure 4.

Intuitively, the loop models a random walk of a particle x that terminates when the particle hits the origin 0. The random walk either with probability ${}^{1}\!/_{x+1}$ immediately terminates or with probability ${}^{x}\!/_{x+1}$ increments the position of x by one. This means that for each iteration where the loop does not terminate, it is even *more likely not to terminate in the next iteration.* Thus, the longer the loop runs, the less likely it will terminate since the probability to continue looping approaches 1 asymptotically. Yet this loop terminates almost-surely, as we will now prove.

[14] Coefficients need to be positive because otherwise $V \geq 0$ cannot be ensured. However, this is not crucial in this proof.

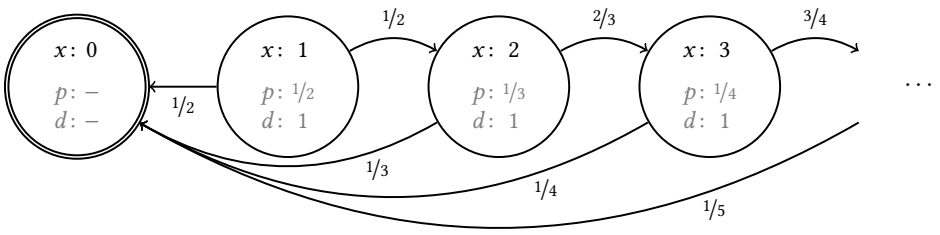

Fig. 4. Execution of the escaping spline loop. The value of the variant is equal to the value of the variable x in each state. Inside the nodes we give the valuations of variable x as well as the values of the probability function p and the decrease function d in each state. Note that in Thm. 4.1 it does not matter what d, p's values are when $V{=}0$, because the *lhs* of (iii) is zero in that case.

Proof of almost-sure termination. The loop guard is given by $G = x{>}0$ and the loop body by

$$C = q \coloneqq 1/x{+}1;\ \{x \coloneqq 0\}\ {}_q\!\oplus\ \{x \coloneqq x+1\}\ .$$

We now prove almost-sure termination of the loop by choosing the standard invariant $I = \mathsf{true}$ and an appropriate p, d and quasi-variant V:

$$V = x, \qquad \text{for } d(v) = 1 \quad \text{and} \quad p(v) = \frac{1}{v+1}\ .$$

Intuitively this tells us that the variant V, i.e. the value of x, decreases with probability at least $1/V{+}1 = 1/x{+}1$ by at least 1 through an iteration of the loop body in case that the guard is satisfied. Now V satisfies the super-martingale property:

$$\begin{aligned}
[G]\cdot \mathsf{awp}\,.\,C\,.\,V &= [x{>}0]\cdot \mathsf{awp}\,.\,q \coloneqq 1/x{+}1;\ \{x \coloneqq 0\}\ {}_q\!\oplus\ \{x \coloneqq x+1\}\,.\,x \\
&= [x{>}0]\cdot \mathsf{awp}\,.\,q \coloneqq 1/x{+}1\,.\,(q\cdot 0 + (1-q)\cdot(x+1)) \\
&= [x{>}0]\cdot\left(1-\frac{1}{x+1}\right)\cdot(x+1) \\
&= [x{>}0]\cdot(x+1-1) \\
&= [x{>}0]\cdot x \\
&\le x \\
&= V\ .
\end{aligned}$$

And V, p, and d satisfy the progress condition for all R:

$$\begin{aligned}
& p(R)\cdot[G\wedge I\wedge x{=}R] \le \mathsf{wp}\,.\,C\,.\,[V\le R-d(R)] \\
\iff\ & \frac{1}{R+1}\cdot[x{>}0\wedge x{=}R] \le \mathsf{wp}\,.\,q \coloneqq 1/x{+}1;\ \{x \coloneqq 0\}\ {}_q\!\oplus\ \{x \coloneqq x+1\}\,.\,[x\le R{-}1] \\
\iff\ & \frac{1}{R+1}\cdot[x{>}0\wedge x{=}R] \le \mathsf{wp}\,.\,q \coloneqq 1/x{+}1\,.\,(q\cdot[0\le R-1] + (1{-}q)\cdot[x{+}1\le R-1])
\end{aligned}$$

$$
\begin{array}{lll}
\iff & \dfrac{1}{R+1} \cdot [x{>}0 \wedge x{=}R] & \leq \ {}^{1}\!/_{x+1} \cdot [0 \leq R-1] + {}^{x}\!/_{x+1} \cdot [x{+}1 \leq R-1] \\
\iff & \dfrac{1}{R+1} \cdot [R{>}0 \wedge x{=}R] & \leq \ {}^{1}\!/_{R+1} \cdot [0 \leq R-1 \wedge x{=}R] + {}^{R}\!/_{R+1} \cdot [R{+}1 \leq R-1 \wedge x{=}R] \\
\iff & \dfrac{1}{R+1} \cdot [R{>}0 \wedge x{=}R] & \leq \ {}^{1}\!/_{R+1} \cdot [0 \leq R-1 \wedge x{=}R] \\
\impliedby & x \in \mathbb{N}\,. & \qquad\qquad\qquad\qquad\qquad\qquad \text{(true by assumption)}
\end{array}
$$

This shows that all preconditions of Theorem 4.1 are satisfied and as a consequence the escaping spline loop terminates almost-surely.

In fact in retrospect *AST* for this loop is not so surprising after all: by inspection, the probability associated with the sole diverging path from say $x{=}1$ is ${}^{1}\!/_{2} \cdot {}^{2}\!/_{3} \cdots = 0$. It is interesting however that this criterion applies in general: if the probability of going up from x is p_x, then the variant $V(x) = {}^{1}\!/_{p_1 p_2 \cdots p_{x-1}}$ is a martingale by construction. And if $p_1 p_2 \cdots > 0$, i.e. the probability of divergence is non-zero, then this variant is bounded and, for reasons discussed below at Cor. 6.2, it therefore acts as a certificate for *non-termination*. Moreover, as illustrated in §8.2, indeed our Thm. 4.1 does not apply when $p_1 p_2 \cdots > 0$ since then there is no everywhere positive but antitone $p()$. [15] If however $p_1 p_2 \cdots = 0$, i.e. the probability of divergence is zero, then the construction $V(x) = {}^{1}\!/_{p_1 p_2 \cdots p_{x-1}}$ works (because the variant is unbounded) — a (limited) completeness property.

6 REVIEW OF MATHEMATICAL LITERATURE ON SUPER-MARTINGALE METHODS

6.1 Recurrent Markov Chains, and Super-Martingales

Early work on characterising recurrent behaviours of infinite-state Markov processes using super-martingale methods is primarily due to Foster [1951, 1952], Kendall [1951] and Blackwell [1955]. In this section we review some of these important results and explain how they relate to *AST* for probabilistic programs and Thm. 4.1. Note that their arguments are given directly in an underlying model of (deterministic) transition systems.

Following the conventions of the authors above, we assume an enumeration of the (countable) state space $i = 0, 1, 2, \ldots$, and transition probabilities p_{ij} for the probability of transitioning from state i to state j. The probability of reaching j from i on the n'th transition is p^n_{ij}, where p^n is computed from single transitions p_{ik} using matrix multiplication. Foster [1951] identified three kinds of long-term average behaviours for infinite-state Markov processes, which behaviours he called dissipative, semi-dissipative and non-dissipative. A process is said to be *non-dissipative* if its long term average behaviour does not "dissipate", i.e. if $\sum_{j \geq 0} \pi_{ij} = 1$ for all states i, where $\pi_{ij} = \lim_{n \to \infty} \frac{1}{n} \sum_{r=1}^{n} p^r_{ij}$ [Kendall 1951]. An illustration of a *dissipative* process is the biased random walk, with an extreme example given by transition probabilities $p_{i(i+1)} = 1$. The non-dissipative condition is more general than *AST*, but the methods used to prove that a process is non-dissipative nevertheless do use super-martingales. In particular Foster's Theorem 5 1951 gives such a sufficient condition for a process to be non-dissipative. It is

$$\sum_{j \geq 0} j{\cdot}p_{ij} \leq i\,, \quad \textit{for all states } i \geq 0\,. \tag{15}$$

Kendall [1951] generalised Foster's (15) by removing the strict relation between the "super-martingale" values and the enumeration of the state space, whilst articulating an important finitary

[15] If ${}^{1}\!/_{p_1 p_2 \cdots} = K < \infty$ then necessarily the escape probabilities $1{-}p(v)$ tend to zero as $V(x){=}v$ tends to K, and so $p(v)$ for any $v{>}K$ must actually *be* zero — which is not allowed, even if the process never reaches x with $V(x){>}K$.

property of a super-martingale that he used in his proof. In Kendall's work, a Markov process is guaranteed to be non-dissipative if there is a function V from states to reals such that

$$\sum_{j\geq 0} V(j)\cdot p_{ij} \leq V(i) \quad \textit{for all states } i{\geq}0 \tag{16}$$

and for each value $\delta{\geq}0$ there are only finitely many states i such that $V(i) \leq \delta$. Finiteness is crucial here: for the dissipative process with $p_{i(i+1)} = 2/3$ and $p_{i(i-1)} = 1/3$ (which we return to in §8.2) we have $V(i) = \pi_{i0}$ satisfies (16) but, of course, in general $\sum_{j\geq 0} \pi_{ij} = \pi_{i0} < 1$, since it can be shown that π_{i0} is the probability of ever reaching 0 from i.

Then Blackwell [1955] further developed the ideas of Foster and Kendall (sketched above) in order to obtain a complete characterisation of Markov-process behaviour in terms of martingales (i.e. exact); some of Blackwell's results can be adapted to work for probabilistic programs generally to provide a certificate to prove *non-AST*. We summarise Blackwell's results here and then show how we can apply them. We continue with the historical notations.

Let C be a subset of the state space, and fix some initial state $\hat{\imath}$. Say that C is *almost closed* (with respect to that $\hat{\imath}$) iff the following conditions hold:

(1) The probability that C is entered infinitely often, as the process takes transitions (initially) starting from $\hat{\imath}$, is strictly greater than zero and
(2) If C is indeed visited infinitely often, starting from $\hat{\imath}$, then eventually the process remains within C permanently.

Say further that a set C is *atomic* iff C does not contain two disjoint almost-closed subsets. Finally, call a Markov process *simple atomic* if it has a single almost-closed atomic set such that once started from $\hat{\imath}$ the process eventually with probability one is trapped in that set. We then have:

Theorem 6.1. *(Corollary of Blackwell's Thm. 2 on p656) [Blackwell 1955]*
A Markov process is simple atomic (as above) just when the only bounded solution of the equation $\sum_{j\geq 0} p_{ij} \cdot V(j) = V(i)$, that is Blackwell's Equation (his 6) stating that V is an exact martingale, is constant for all i in $S\backslash C$ and transitions p_{ij}. □

We now show how to apply Thm. 6.1 to general probabilistic programs to obtain a certificate for non-termination.

Corollary 6.2 (Non-termination certificate). *We use the conventions of Thm. 4.1, restated here. Let $I, G \subseteq \Sigma$ be predicates; let $V : \Sigma{\to}\mathbb{R}_{\geq 0}$ be a non-negative real-valued function on the state; and let Com be a pGCL program. Then the conditions*

(i) I is a standard invariant for the loop $\texttt{while}\,(G)\,\{Com\}$ *, and*
(ii) $G \wedge I \Rightarrow V{>}0$, and
(iii) V is a non-constant and bounded exact *martingale on $I \wedge G$*

together imply that there is a state σ in I such that $\texttt{wp.while}\,(G)\,\{Com\}.1(\sigma) < 1$. *That is*

> *If a predicate I is a standard invariant, and there exists a non-negative real-valued variant function V on the state, an exact martingale on $I \wedge G$, such that V is bounded and non-constant, then there is some initial state satisfying I from which loop* $\texttt{while}\,(G)\,\{Com\}$ *does not terminate AS.*

Proof. Fix a starting state $\hat{s}$, and collapse the termination set S_0 (i.e. all states that do not satisfy the guard) to a single state s_0. Now adjust the underlying transition system corresponding to the given program so that any transition to a state in S_0 becomes a transition into s_0, and assume that there is a single transition from s_0 to s_0. Suppose now that the probability of $\hat{s}$'s reaching s_0 is one. We now note:

(1) Our termination set $\{s_0\}$ is almost-closed and atomic (in the sense of Blackwell), because
   (a) almost closed: Our process reaches s_0 with non-zero probability (in fact we assumed with probability one, for a contradiction) and, once at s_0, it remains there.
   (b) atomic: Our set $\{s_0\}$ has no non-empty subsets.
(2) We now recall that in fact s_0 is reached with probability one, so that the whole process is simple atomic.
(3) From Blackwell's Thm. 6.1 we conclude that the only possible non-trivial martingale is unbounded.

We deduce therefore, that if there exists a non-constant bounded martingale then there is some state from which termination is not guaranteed with probability 1. □

Thus –in summary– we have specialised Blackwell's result to demonstrate a new refutation certificate for programs: if the martingale is finite and non-constant it actually refutes termination with probability 1, not just finite expected time to termination.

In fact Cor. 6.2 provides an interesting embellishment to recent work by Chatterjee et al. [2017] who introduce the notion of "repulsing super-martingales". Their *Theorem 6* uses an ε-repulsing super-martingale with $\varepsilon{>}0$ to refute almost-sure termination. And their *Theorem 7* uses an ε repulsing super-martingale with $\varepsilon{\geq}0$ to refute finite expected time to termination. In particular to refute finite expected time to termination only a martingale is required.

Our Cor. 6.2 takes this further to use non-constant and bounded martingales as certificates to refute almost-sure termination. For example the one-dimensional random walker

$$\texttt{while}\,(x>0)\,\{\{x:=x-1\}\,{}_{1/2}\!\oplus\,\{x:=x+1\}\}$$

has an exact *unbounded* martingale, and therefore our rule Thm. 4.1 shows that it terminates with probability 1. On the other hand the *biased* walker $\texttt{while}\,(x>0)\,\{\{x:=x-1\}\,{}_{1/3}\!\oplus\,\{x:=x+1\}\}$ (from §8.2) has a *non-constant* and *bounded* martingale based on the function $V(s) = 1{-}\pi_{s\rightsquigarrow 0}$ where $\pi_{s\rightsquigarrow 0}$ is the probability that, starting from state s, eventually state 0 (i.e. $x{=}0$) is reached . By Cor. 6.2 we can conclude that the program does not terminate with probability 1. Note that Chatterjee's Theorem 7 2017 does not distinguish between these two cases in terms of their behaviour: it implies that neither has finite expected time to terminate. And Cor. 6.2 holds even when demonic choice is present.

6.2 Towards Completeness: The Case of the Random Walker in Two Dimensions

Foster [1952] further considers the question of conditions on a Markov process that imply the existence of a super-martingale; this is relevant for our Theme C. His conditions are:

(1) The state space Σ is countable;
(2) There is a finite subset $C \subseteq \Sigma$ that is reached with probability 1 from any other state;
(3) The states are numbered so that given any pair of states s_i, s_j there is some probability of reaching s_j from s_i whenever $i{<}j$;
(4) There is a single probability $0{<}\delta{<}1$ for the whole system such that for any N there is an i such that for all $j{\geq}i$ the state s_j cannot reach C within N steps and with probability at least δ.

Under these conditions, Foster shows that there exists an *unbounded* super-martingale function V on S such that $V(s)$ tends to infinity as the numbering of s tends to infinity.

The construction is a variation on the expected time to termination but, as he remarks, expected time cannot be used because in many situations the expected time to termination is infinite. However using Foster's construction we can prove the existence of a super-martingale that also satisfies the progress conditions of our rule Thm. 4.1, and thus could be used to prove termination for the 2-dimensional symmetric random walk

$$\texttt{while}\,(x{\neq}0 \vee y{\neq}0)\,\{x \coloneqq x{-}1 \;\oplus\; x \coloneqq x{+}1 \;\oplus\; y \coloneqq y{-}1 \;\oplus\; y \coloneqq y{+}1\}$$

where iterated $\oplus$ is shorthand for uniform choice (in this case $1/4$ each).

Corollary 6.3 (Two-dimensional random walk). *There exists a super-martingale which satisfies the conditions of Thm. 4.1 to prove termination of the two-dimensional random walker.*

Proof. (Sketch.) We follow Foster's argument 1952 to show that there is a numbering of the states that satisfy his conditions for constructing a super-martingale; then we show that the constructed super-martingale also satisfies the progress conditions. Foster enumerates the states by "spiralling out" through increasing Manhattan distance, observing that simple scheme to satisfy his enumeration conditions. Then he shows that there is a variant function V which satisfies the conditions for a super-martingale; [16] and in fact as the numbering of s approaches infinity so too does $V(s)$; in particular Foster shows that there are no accumulation points in the image of V. Foster's general proof is by construction. (We sketch it in App. F.)

To show that our rule Thm. 4.1 applies, we need however to establish a progress condition. First define $p(v)$ to be $1/4$ for all v. Then for d, first consider the subset $S_{\leq v}$ of S comprising all those s with $V(s){\leq}v$. Because there are no accumulation points in the image of V, we must have that $S_{\leq v}$ is finite. Now set $d(v)$ to be the minimum non-zero distance between any two of them, that is $(min\ (V(s'){-}V(s)) \mid s, s' \in (S_{\leq v}) \wedge V(s'){>}V(s))$. Since $V(s)$ increases arbitrarily we have that d is non-zero whenever $v{=}V(s)$ for some state with Manhattan distance strictly greater than 0.

Thus there is guaranteed to be a V satisfying the progress condition Thm. 4.1(iii) that establishes termination for the 2dSRW — even if we don't know what it is in closed form. □

7 REVIEW OF RELATED WORK ON TERMINATION FOR PROBABILISTIC PROGRAMS

Our earlier variant rule Thm. 3.4 [Morgan 1996, Sec. 6],[McIver and Morgan 2005, Sec. 2.7] effectively made p, d constants, imposed no super-martingale condition but instead bounded V above, making it not sufficient for the random walk. Later however we did prove the symmetric random walk to be *AST* using a rule more like the current one [McIver and Morgan 2005, Sec. 3.3].

Chakarov and Sankaranarayanan [2013] consider the use of martingales for the analysis of infinite-state probabilistic programs, and Chakarov [2016] has done further, more extensive work.

Chakarov and Sankaranarayanan also show that a *ranking super-martingale* implies *AST*, and a key property of their definition for ranking super-martingale is that there is some constant $\varepsilon{>}0$ such that the average decrease of the super-martingale is everywhere (except for the termination states) at least ε. Their program model is operates over discrete distributions, without nondeterminism.

That work is an important step towards applying results from probability theory to the verification of infinite-state probabilistic programs.

Ferrer Fioriti and Hermanns [2015] also use ranking super-martingales, with results that provide a significant extension to Chakarov and Sankaranarayanan's work [Chakarov and Sankaranarayanan 2013]. Their program model includes both non-determinism and continuous probability

[16] The Manhattan distance itself is not a super-martingale because, on the axes, the distance actually *increases* in expectation by $(-1 + 1 + 1 + 1)/4 = 1/2$. Indeed if the Manhattan variant worked for two dimensions, it would also work for three; but the 3dSRW is not *AST*.

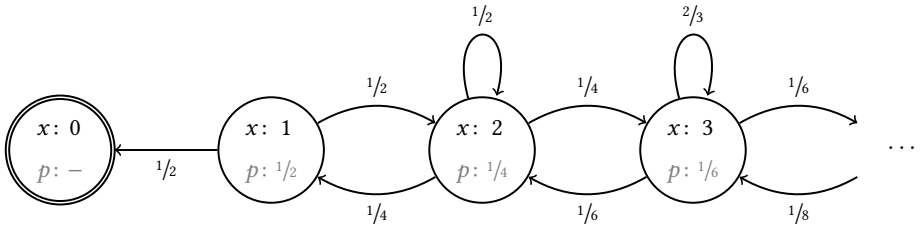

Fig. 5. Transition system for the Lazy Loper program (above, and App. D). Inside the nodes we give the valuations of variable x as well as the value of the probability function p. The value of the variant is equal to the value of variable x in each state. The value of the decrease function d is constantly 1.

distributions over transitions. They also show completeness for the class of programs whose expected time to termination is finite. That excludes the random walk however; but they do demonstrate by example that the method can still apply to some systems which do not have finite termination time.

We note that it can be shown that a ranking super-martingale that proves *AS* also satisfies p, d progress for Thm. 4.1; see App. G.

Chatterjee et al. [2017] study techniques for proving that programs terminate with some probability (not necessarily one). Their innovation is to introduce the concept of "repulsing super-martingales" — these are also super-martingales with values that decrease outside of some defined set. Repulsing super-martingales can obtain lower bounds on termination probabilities, and as certificates can refute almost-sure termination and finite expected times to termination.

More recently still **Chatterjee and Fu [2017]** have studied termination for probabilistic and non-deterministic recursive programs. In particular they show that "conditionally difference-bounded ranking super-martingales" can be used to prove almost-sure termination. As we do, Chatterjee and Fu allow super-martingales (i.e. not necessarily ranking); and their Thm. 5 requires that the average absolute difference between $V(\sigma)$ and $V(\sigma')$ must be at least some fixed $\delta{>}0$. This constraint seems to imply some kind of progress and it will be an interesting exercise to understand exactly the differences in applicability between the two rules. For example the existence of a fixed $\delta{>}0$ allows Chatterjee and Fu to give an estimate for "tail probabilities". On the other hand the variation of the random walker given by the "Lazy Loper" program of Fig. 5, that is

$$\texttt{while}\,(x > 0)\,\{\quad \{\{x := x{+}1\}\ {}_{1/2}\!\oplus\ \{x := x{-}1\}\}\ {}_{1/x}\!\oplus\ \{\mathit{Skip}\}\quad \}$$

in which the walker "dawdles" at a location depending on the distance to the origin, nevertheless can be proved to terminate almost surely using Thm. 4.1 with definitions $V(x) = x$, and $p(v) = 1 \min 1/2v$ and $d(v) = 1$ for progress; but Chatterjee's Thm. 5 2017 does not seem to apply here. Moreover there appears to be no super-martingale for this program that has average absolute move bounded away from 0. We give details in App. D.

Finally, **Agrawal et al. [2018]** have extended the ε-strict super-martingale approach to include lexicographic orderings, and present techniques for their automatic synthesis. (We explore parametrised-ε super-martingales, but not lexicographic, in McIver and Morgan [2016, Sec. 5].)

A different approach to the same issue is the work of **Lago and Grellois [2017]** in which expressions themselves are probabilistic artefacts, and their termination properties can be "inherited" by functional programs containing them: that allows the expressions' behaviour to be studied separately, outside of the clutter of the program containing them.

There are a number of other works that demonstrate tool support based on the above and similar techniques. All the authors above [Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2017; Ferrer Fioriti and Hermanns 2015] have developed and implemented algorithms to support verification based on super-martingales. **Esparza et al. [2012]** develop algorithmic support for *AST* of "weakly finite" programs, where a program is *weakly finite* if the set of states reachable from any initial state is finite. **Kaminski et al. [2016]** have studied the analysis of expected termination times of infinite state systems using probabilistic invariant-style reasoning, with some applications to *AST*. In even earlier work **Celiku and McIver [2005]** explore the mechanisation of upper bounds on expected termination times, taking probabilistic weakest pre-expectations [McIver and Morgan 2005] for their model of probability and non-determinism.

8 THEORETICAL ISSUES, LIMITATIONS AND CAVEATS

8.1 How Much Nondeterminism?

Our arguments above are over "expectation transformers", i.e. functions from post-expectations to pre-expectations and thus going in effect "backwards". But equivalently our programs are functions from initial state to (discrete) distribution over final states or, when demonic choice is present, to *sets* of such distributions (but only sets satisfying certain "healthiness" conditions). That equivalence was shown by Kozen [1985] for deterministic (i.e. non-demonic) programs, and extended by McIver and Morgan [2005]; Morgan et al. [1996] when demonic choice was added. Table 1 interprets programs (syntax) into that semantic space, and e.g. Thm. 3.4 and Thm. 3.5, crucial to our argument, have been shown to be true in that space [McIver and Morgan 2005].

Important is that those two theorems were *not* proved by structural induction over pGCL syntax directly; rather they follow from a different structurally inductive proof, that all pGCL programs are mapped into the semantic space (where the theorems hold) — that is, a proof that the space is closed under program-combining operators. The significance of the difference is that our results therefore hold for any elements of that space, whether they come from pGCL or not, including operational descriptions of programs as transition systems provided they satisfy the healthiness conditions the space demands. One such condition is the restriction to discrete distributions. [17]

Another healthiness condition concerns the degree of demonic choice our semantic space allows: is it finite? countable? unlimited? In fact our space requires that the sets of distributions be closed in the product topology over the set of discrete (sub-)distributions on Σ, that is distributions whose total weight is *no more than* 1. (Any missing weight indicates non-termination.) All (meanings) of pGCL programs have that property [McIver and Morgan 2005]; and all *finitely* branching transition systems do. But that property is not the same e.g. as countable vs. uncountable branching. For example, Program

$$c, x \coloneqq \mathit{true}, 0; \quad \texttt{while}\,(c)\,\{\{c \coloneqq \mathit{false}\}\ {}_{1/2}\!\oplus\ \{x \coloneqq x+2\}\}; \quad \{x \coloneqq x+1\}\ \Box\ \{\texttt{skip}\} \tag{17}$$

makes *uncountably* many demonic choices (over geometric-style discrete distributions). [18] Nevertheless, because the program is written in pGCL, that set is closed. On the other hand, the (standard) program "choose n from the natural numbers" has only *countably infinite* branching, and yet

[17] Thus e.g. part of the structurally inductive proof would be to show that loops with discrete-distribution bodies cannot somehow "in the limit" require a proper measure to define their overall effect: the worst it can get is a countably infinite but still discrete distribution.

[18] First pick any real number b in the unit interval $[0, 1]$ (which action cannot be written using pGCL's only-binary demonic choice); consider its binary expansion $0.b_1b_2\cdots b_n\cdots$. Construct the discrete (countably infinite) distribution $0{+}b_1$ @1/2, $2{+}b_2$ @1/4, $\cdots$ $2n{+}b_n$ @$1/2^n$ $\cdots$ where "@" means "with probability". (That second step can be done using pGCL, for already-determined b.) For every b chosen in the first step, the above distribution is a possible result, different for each b and so uncountably numerous. But still the set of them all is closed, since the pGCL (17) produces it.

cannot be written in the pGCL of Table 1. Embedded in the probabilistic model [McIver and Morgan 2005], its output set of distributions is not closed — and so this program is out-of-scope for us. But Program (17) is within our scope.

Thus the conceptual boundary of our result is *not* countable vs. uncountable branching: rather it is topological closure vs. non-closure of sets of discrete distributions. But this issue is important only for examples "imported" from outside of pGCL; for any pGCL program, closure of the corresponding transition system's results sets is automatic [McIver and Morgan 2005, Sec. 8.2].

8.2 "Progress" is More Demanding than it Looks

Consider the asymmetric random walker $x \coloneqq 1;\ \texttt{while}\,(x{\neq}0)\,\{\{x \coloneqq x{-}1\}\ {}_{1/3}\!\oplus\ \{x \coloneqq x{+}1\}\}$. We can easily synthesise an exact- (and thus super-) martingale $V(x) = {}^{2^x-1}\!/_{2^{x-1}}$ by solving the associated recurrence. It is bounded asymptotically above by 2, so that for progress we are tempted by $p(v){=}1/3$ and $d(v) = 2{-}v$, both satisfying our positive-and-antitone requirements when $v{<}2$.

But this $d()$ in fact does not satisfy our requirements, because they apply for *all* v, not just those generated by states that the program can actually reach. And in this case there is no suitable value for $d(2)$, since it would have to be 0 for d to be antitone. That is, even though the program can never reach a state x where $V(x){=}2$, the requirements on $d(2)$ still apply.

As well as saving us from unsoundness (since the that asymmetric walker is not *AST*), this exposes an important methodological issue: the properties of p, d, their being non-zero and antitone, *do not refer to the program text at all.* However the properties of those functions might be proved –by hand, or with Mathematica or Sage– the semantics of pGCL is not required: one needs only analytic arguments over the reals. And those arguments can be delegated to other people who have never heard of pGCL or transition systems, or Markov processes, random variables or program termination. That is, if we want to use powerful external analytical tools, we should avoid as far as possible that they must be "taught" our semantics.

8.3 Why Do we Express V's Being a Super-Martingale by Writing a Sub-Martingale Inequality?

In Thm. 4.1 we wrote the super-martingale property of V as a sub-martingale property of $H{\ominus}V$; yet in §5, the case studies, we introduce the "angelic" awp and check the super-martingale property directly. Why didn't we use awp in Thm. 4.1 in the first place?

The reason is that Thm. 3.5 is proved over the semantic space of McIver and Morgan [2005] mentioned in §8.1 above, and the brief treatment of angelic choice there [*op. cit.*, Sec. 8.5] gives no awp-based results for loops. To refer to the literature in its own terms –and to avoid building new special-purpose semantics here– we therefore must use only wp when importing existing results.

On the other hand, the equivalence introduced for convenience in §5 –and whose property (11) is established by structural induction over straight-line programs– is used for (12) only and does not rely on closure, or any other sophisticated property of the semantic space.

8.4 Bounded Expectations

In the symmetric random walk on naturals x, the expectation x is an exact martingale in fact; and that process terminates *AS*. If however we had used unbounded x as *Sub* in Thm. 3.5, we could conclude that the expected final value of x is at least the (exact) initial value of x. If the process started at $x{=}1$, therefore, we would conclude that its expected value on termination is at least 1; but we know that its x's expected (in fact exact) value on termination is 0 — a contradiction.

That is why one assumption of Thm. 3.5 is that *Sub* is bounded, and is one reason that, instead of using the potentially unbounded V, we use the bounded $H{\ominus}V$ instead. (See also App. C.)

9 CONCLUSION

We have investigated "parametric" super-martingale methods for proving almost-sure termination for probabilistic- and demonic programs, and our main result Thm. 4.1 presents a new method, described earlier by McIver and Morgan [2016] over a transition system, but now expressed and proved in the probabilistic programming logic of pGCL; the rule can therefore be applied at the source level. Although our presentation is in terms of wp-style reasoning, our innovation of parametrised p, d progress is also applicable to transition-style models of programs. (See, for example Gretz et al.'s interpretation 2014 of wp in terms of explicit transition systems.)

Our rule seems to be able to prove some tricky cases that go beyond other published rules, and moreover we have shown that p, d progress can also be used as alternatives to rules based on ranking super-martingales, and rules based on conditional absolute difference. Furthermore, we believe our rule suffices for the two-dimensional symmetric random walk (§6.2).

Completeness remains an open problem however, although the mathematical literature provides some insight to its solution in certain cases [Blackwell 1955; Foster 1952].

APPENDICES [19]

ACKNOWLEDGMENTS

McIver and Morgan are grateful to David Basin and the Information Security Group at ETH Zürich for hosting a six-month stay in Switzerland, during part of which this work began. And thanks particularly to Andreas Lochbihler, who shared with us the probabilistic termination problem that led to it. They acknowledge the support of ARC grant DP140101119.

Part of this work was carried out during the Workshop on Probabilistic Programming Semantics at McGill University's Bellairs Research Institute on Barbados organised by Alexandra Silva and Prakash Panangaden.

Kaminski and Katoen are grateful to Sebastian Junges for spotting a flaw in §5.4.

REFERENCES

Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. 2018. Lexicographic Ranking Supermartingales: An Efficient Approach to Termination of Probabilistic Programs. In *Proceedings of the 45th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2018)*. ACM, New York, NY, USA.

David Blackwell. 1955. On Transient Markov Processes with a Countable Number of States and Stationary Transition Probabilities. *Ann. Math. Statist.* 26 (1955), 654–658.

Orieta Celiku and Annabelle McIver. 2005. Compositional Specification and Analysis of Cost-Based Properties in Probabilistic Programs. In *FM (Lecture Notes in Computer Science)*, Vol. 3582. Springer, 107–122.

Aleksandar Chakarov. 2016. *Deductive Verification of Infinite-State Stochastic Systems using Martingales.* Ph.D. Dissertation. University of Colorado at Boulder.

Aleksandar Chakarov and Sriram Sankaranarayanan. 2013. Probabilistic Program Analysis with Martingales. In *CAV (Lecture Notes in Computer Science)*, Vol. 8044. Springer, 511–526.

Krishnendu Chatterjee and Hongfei Fu. 2017. Termination of Nondeterministic Recursive Probabilistic Programs. *CoRR* abs/1701.02944 (2017).

Krishnendu Chatterjee, Petr Novotný, and Dorde Žikelić. 2017. Stochastic Invariants for Probabilistic Termination. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017)*. ACM, New York, NY, USA, 145–160. https://doi.org/10.1145/3009837.3009873

Edsger W. Dijkstra. 1976. *A Discipline of Programming.* Prentice-Hall.

Javier Esparza, Andreas Gaiser, and Stefan Kiefer. 2012. Proving Termination of Probabilistic Programs Using Patterns. In *CAV (Lecture Notes in Computer Science)*, Vol. 7358. Springer, 123–138.

Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2015)*. ACM, New York, NY, USA, 489–501. https://doi.org/10.1145/2676726.2677001

[19] Appendices may be found in *arXiv*.

F. G. Foster. 1951. Markoff chains with an enumerable number of states and a class of cascade processes. *Cambridge Philosophical Society* 1, 47 (1951), 77–85.

F. G. Foster. 1952. On Markov Chains with an Enumerable Infinity of States. *Mathematical Proceedings of the Cambridge Philosophical Society* 4 (Oct 1952), 587–591. https://doi.org/10.1017/S0305004100076362

Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. 2014. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Perform. Eval.* 73 (2014), 110–132.

G.R. Grimmett and D. Welsh. 1986. *Probability: an Introduction.* Oxford Science Publications.

Sergiu Hart, Micha Sharir, and Amir Pnueli. 1983. Termination of Probabilistic Concurrent Programs. *ACM Trans. Program. Lang. Syst.* 5, 3 (July 1983), 356–380. https://doi.org/10.1145/2166.357214

C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (1969), 576–580.

Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2016. Weakest Precondition Reasoning for Expected Run-Times of Probabilistic Programs. In *ESOP (Lecture Notes in Computer Science)*, Vol. 9632. Springer, 364–389.

David G. Kendall. 1951. On non-dissipative Markoff chains with an enumerable infinity of states. *Mathematical Proceedings of the Cambridge Philosophical Society* 47, 3 (001 007 1951), 633–634. https://doi.org/10.1017/S0305004100027055

Konrad Knopp. 1928. *Theory and Application of Infinite Series.* London.

Dexter Kozen. 1985. A Probabilistic PDL. *J. Comput. Syst. Sci.* 30, 2 (1985), 162–178.

Ugo Dal Lago and Charles Grellois. 2017. Probabilistic Termination by Monadic Affine Sized Typing. In *ESOP (Lecture Notes in Computer Science)*, Vol. 10201. Springer, 393–419.

Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems.* Springer.

Annabelle McIver and Carroll Morgan. 2016. A New Rule for Almost-Certain Termination of Probabilistic and Demonic Programs. *CoRR* abs/1612.01091 (2016).

C.C. Morgan. 1996. Proof Rules for Probabilistic Loops. In *Proc BCS-FACS 7th Refinement Workshop (Workshops in Computing)*, He Jifeng, John Cooke, and Peter Wallis (Eds.). Springer. `http://www.bcs.org/upload/pdf/ewic_rw96_paper10.pdf`.

Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic Predicate Transformers. *ACM Trans. Program. Lang. Syst.* 18, 3 (May 1996), 325–353. https://doi.org/10.1145/229542.229547

Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning About Recursive Probabilistic Programs. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '16)*. ACM, New York, NY, USA, 672–681. https://doi.org/10.1145/2933575.2935317