SIAM J. COMPUT.
Vol. 13, No. 2, May 1984

006

VERIFICATION OF PROBABILISTIC PROGRAMS*

MICHA SHARIR†, AMIR PNUELI‡ AND SERGIU HART§

Abstract. A general method for proving properties of probabilistic programs is presented. This method generalizes the intermediate assertion method in that it extends a given assertion on the output distribution into an invariant assertion on all intermediate distributions, too. The proof method is shown to be sound and complete for programs which terminate with probability 1. A dual approach, based on the expected number of visits in each intermediate state, is also presented. All the methods are presented under the uniform framework which considers a probabilistic program as a discrete Markov process.

Key words. program verification, probabilistic programs, Markov chains

CR categories. 5.24, 5.5

Introduction. In this work we examine the possibility of developing verification methodology for probabilistic programs. The need for analysis of probabilistic programs arises in two main situations. The first is when we analyze a deterministic program whose inputs are drawn out of a space with some known probability distribution, and we wish to infer some statistical property of the program, such as its average running time, the expected value of some output variable, the probability of program termination, etc. Another situation is that of a nondeterministic program where the decision in nondeterministic forks in the program is made according to some known distribution. We could have, of course, a combination of the two where both the input values and nondeterministic choices within the program are chosen at random according to known distributions.

With the recent emergence of probabilistic algorithms, such as primality testing [RB] and synchronization between concurrent processes [LR], and the more conventional problem of average behavior of deterministic algorithms, the need for tools for probabilistic verification becomes increasingly urgent.

One possible approach to the probabilistic analysis of programs, which must certainly be the first step towards any coherent theory of the subject, is the definition of the probabilistic semantics of programs. Such an approach is taken for example in [KO] where a probabilistic program is regarded as a distribution transformer, transforming an input distribution into an output distribution. The output distribution then tells us the probability for the program to terminate in any of its terminal states. In principle, once we know how to compute the probability of each terminal state, we have captured the complete (input/output) behavior of the program, and each specific question can be settled by referring to the output distribution. In practice, however, when we are interested in a specific question, the computation of the complete distribution transformation is often a formidable and unnecessary task. This is why, in the nonprobabilistic case, the disciplines of semantic assignment and verification are closely related but still separate. The first seeks to define the mathematical interpretation of programs in a given language. The latter tries to offer methods by which specific questions about a program can be vigorously settled by extracting from

* Received by the editors October 28, 1981, and in revised form January 21, 1983.

† Department of Computer Science, Tel Aviv University, Tel Aviv, Israel. The research of this author was supported partly by the Office of Naval Research under grant N00014-75-C-0571, while the author was visiting Courant Institute, New York University, and partly by the Bat-Sheva Fund, Israel.

‡ Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel.

§ Department of Statistics, Tel Aviv University, Tel Aviv, Israel.

the program just the *minimal* amount of information which is required in order to settle the question. In one sense the theory of verification can be regarded as the theory of semantic approximation.

Taking as our starting point the probabilistic semantics of programs, as defined for example in [KO], we set out to see whether the verification methods that proved successful in the deterministic case, such as the intermediate assertion method [FL], computational induction [PA] and subgoal induction [MW], can be generalized to the probabilistic case.

The salient features of all these methods are:

a) They are *goal oriented*; i.e., the verification conditions to be solved depend on the property to be proved, and we only work so hard as is needed in order to establish the particular property.

b) The verification conditions are *local* in the sense that they connect two consecutive instants in the execution of the program.

c) If we insist on the *minimal* solution to the verification conditions we come up with the full semantics of the program, or an equivalent characterization. These are for example the minimal invariant predicates in Floyd's method.

As will be shown below, we suggest two generalizations. The first is an extension of the intermediate assertion method with some of the flavor of subgoal induction. Starting with an assertion on the terminal states which is supposed to hold upon termination, we seek to extend it into an assertion on all the states which holds continuously throughout the execution. The second method is similar to computational induction. We form equations which express changes in the distribution due to a single program step. The minimal solution to these equations gives exactly the terminal distribution. Consequently every solution, not necessarily the minimal, provides an upper bound to the terminal distribution.

We show that these two approaches are dual in the sense that they are both derived from the same matrix describing the program, and both obey certain duality relationships which allow us to combine the information yielded by each approach separately.

The methods are presented in a uniform framework which considers programs as global state transformations. It should not be too difficult to adapt them to more structured representations of programs. Specifically: The basic approach treats a probabilistic program as a Markov process that goes by a chain of transitions through the program states. At each step, depending on the current state, there are known probabilities for the next state, and the process chooses the next state according to their distribution.

These probabilities depend on the nature of the program statement about to be executed. If this statement is not a random draw, then there is a unique next state; otherwise, there may be several succeeding states, depending on the outcome of the draw made by the program. Thus, in such a model the state-transition probabilities can be assumed to be given a priority.

Note that random distribution of the program's input will manifest itself in similar distribution of the program states, but not in the transition probabilities among them.

Since, in principle, Markov chains model infinite processes, our approach can therefore also assign semantics to nonterminating probabilistic programs. This generalizes other approaches based on input–output semantics, which ignores nonterminating executions (see [KO] for example). To deal with terminating programs, we regard the terminal program states as absorbing states which once in them the process can never escape.

This paper is organized as follows. In § 1, we define probabilistic programs as Markov chains and assign to them semantics defined in terms of certain well-known quantities associated with such chains. We show that these semantics coincide with the semantics defined by Kozen [KO] and generalize it to nonstructured programs. In § 2, we describe our first method of probabilistic verification, which is based on invariant functionals on the program states' distribution. In § 3, we describe a second verification method based on the expected number of visits in the nonterminating states. Essentially this approach had been formerly suggested by Ramshaw [RA], but under a different framework. When cast into the framework of Markov chains, the approach becomes greatly simplified and much of the theory developed by Ramshaw turns out to be straightforward consequences of Markov chain theory. We also establish some "duality" relationships between our two approaches. In § 4, we demonstrate our verification methods in a series of examples.

When considering probabilistic programs as Markov chains, it is important to bear in mind that this representation is faithful only if all the data that can affect program execution is incorporated into the program's (or rather the chain's) states. Thus, if the program execution depends largely on its input, which in turn is drawn from some complex distribution, then incorporating the whole input data into the program states may make its analysis as a Markov chain rather difficult. (For example, if the program itself is fully deterministic, it may well be the case that its Markov chain representation decomposes into many disjoint chains, one per each input value.) The Markov chain representation is most favorable in cases where the probabilistic nature of the program arises from random draws made by the program itself.

Quite surprisingly, very few researchers have used the Markov chain model to represent probabilistic programs, although a completely static treatment of programs as Markov chains (with states being program locations only) has long been suggested by [RM]. Saheb-Djahromi [SD] uses Markov chains to define the operational semantics of a probabilistic version of the language LCF. The works of Kozen [KO] and Ramshaw [RA] mentioned above do not use Markov chains, although most of their results can be easily interpreted as standard results in Markov chain theory. Probabilistic analysis of programs is also studied by Wegbreit [WE].

Recently, the authors have generalized analysis of probabilistic programs to the case of concurrent programs [HSP], [HS]. Their program execution can be described as a certain cooperation of several Markov chains, and its analysis requires special techniques, unlike the classical Markov chain theory used in this paper. Another interesting and recent direction of research is the development of probabilistic logics for reasoning about properties of probabilistic programs (cf. [RE], [LS], [HS2]).

1. Probabilistic programs and their semantics. In our framework, a program is considered as a (probabilistic) transformation operating on a set of states. Let S denote the set of program states which may be infinite but countable. (See, however, § 4 for a treatment of uncountably many states.) We assume that the action of a single step of the program is represented by a given matrix of transition probabilities $P = \{P_{ij}\}$. Thus P_{ij} is the probability of going from state $i \in S$ to state $j \in S$ in one step. Let $\bar{\mu}^0$ be the initial distribution vector which specifies for each state $i \in S$ the probability $\mu_i^0 \geqq 0$ that initially the program is in this state. Based on the assumption that the probability for a transition from state i to state j depends only on i and j (and not on the time or any other nonlocal entity), an execution of a probabilistic program can be regarded as a Markov process which goes through a chain of discrete S states [CH], [RV], [KSK].

To illustrate these concepts, consider the following program:

$i := 0;$
l_1: **while** random $(p\delta_0 + q\delta_1) = 0$ **do** $i := i + 1;$
l_2: **halt**.

In general, the probabilistic expression "random (λ)" chooses a random value according to the distribution λ. In this case δ_i is a unit distribution concentrated at i, $i = 0, 1$. Thus, for $p + q = 1$, random$(p\delta_0 + q\delta_1)$ chooses 0 with probability p and 1 with probability $q = 1 - p$. Ignoring the initializing step, the set of states for this program is:

$$S = \{(l_1, i), (l_2, i) | i \geqq 0\}.$$

Note that states include the location in the program as well as values for all the program variables.

The initial distribution is given by:

$$\mu^0_{(l_1,i)} = \delta_{i,0}, \qquad \mu^0_{(l_2,i)} = 0.$$

That is: with certainty the initial state is $(l_1, 0)$. The transition probabilities are given by:

$$P_{(l_1,i)(l_2,i)} = q, \qquad P_{(l_1,i)(l_1,i+1)} = p, \qquad i \geqq 0.$$
$$P_{(l_2,i)(l_2,i)} = 1,$$

All other transitions have probability 0.

We partition our state space $S = I \cup T$. The set T is the set of all terminal states (absorbing states); for each $t \in T$, $P_{ts} = \delta_{t,s}$; i.e., with certainty we remain at t for the next stage and hence forever. The set $I = S - T$ is the set of intermediate states. Thus in the example above,

$$I = \{(l_1, i) | i \geqq 0\}, \qquad T = \{(l_2, i) | i \geqq 0\}.$$

Let us define:

$$P^{(n)}_{ij} = \{\text{Probability of reaching state } j \text{ from state } i \text{ in exactly } n \text{ steps}\}.$$

Obviously $P^{(n)}_{ij} = \{P^n\}_{i,j}$ where P^n is the nth power of the (infinite) transition probability matrix P. This can also be written as:

$$P^{(n)}_{ij} = \sum P_{ii_1} \cdot P_{i_1 i_2} \cdot \cdots \cdot P_{i_{n-1} j}$$

where the summation extends over all $(n-1)$-tuples $(i_1, \cdots, i_{n-1})$. Corresponding to an initial distribution $\bar{\mu}^0$, we can also define:

$$\mu^{(n)}_j = \{\text{Probability of being in state } j \text{ after } n \geqq 0 \text{ steps}\}.$$

Obviously $\mu^{(n)}_j = \sum_{i \in S} \mu^0_i P^{(n)}_{ij}$, or in matrix notation $\bar{\mu}^{(n)} = \bar{\mu}^0 P^n$. Note that, since each $j \in T$ is an absorbing state, the sequence $\{\mu^{(n)}_j\}_{n \geqq 0}$ is nondecreasing for each $j \in T$.

We also define:

$f^{(n)}_{ij} = \{\text{Probability of reaching state } j \text{ from state } i \textit{ for the first time} \text{ in exactly } n \text{ steps}\}.$

These quantities satisfy:

$$f^{(n)}_{ij} = \sum P_{ii_1} \cdot \cdots \cdot P_{i_{n-1} j}$$

where the summation extends over all $(n-1)$-tuples $(i_1, \cdots, i_{n-1})$ of states all of which are different from j.

Clearly $f_{ij}^{(n)}$ or $\mu_j^{(n)}$, given an initial distribution, fully describe the behavior of the program. However, they are too detailed, and we would like to take out the dependence on the step counter n. One such integrated measure is given by:

$$f_{ij}^* = \sum_{n=1}^{\infty} f_{ij}^{(n)}$$

where f_{ij}^* is the probability of *ever* getting to state j from state i.

Similarly we define μ_j^* as the probability of ever getting to state j, given that the initial distribution is $\bar{\mu}^0$. Obviously:

$$\mu_j^* = \sum_{i \in S} \mu_i^0 f_{ij}^*. \tag{1}$$

If we restrict ourselves to terminal states $j \in T$, then since j is an absorbing state it follows that:

$$f_{ij}^* = \lim_{n \to \infty} P_{ij}^{(n)} \quad \text{and} \quad \mu_j^* = \lim_{n \to \infty} \mu_j^{(n)} \quad \text{for } j \in T.$$

The f_{ij}^* for $j \in T$ can be considered as the input–output semantics of the program viewed as a distribution transformer, in that given an initial distribution $\bar{\mu}^0$ the terminal distribution is given by $\bar{\mu}^* = \bar{\mu}^0 F^*$ where $F^* = \{f_{ij}^*\}$.

We will therefore regard the program as being fully specified when the matrix F^* is given. It should be noted that this is a generalization of Kozen's semantics, provided that one restricts oneself to discrete distributions. Note that Kozen defines the semantics of only a restricted class of structured programs, whereas our interpretation does not impose any such restriction. Let us indeed compare the two approaches for a while loop of the form

while $x \in B$ **do** Q.

For simplicity, let us identify the program states with values of x. The terminating states are then elements of B^c. Suppose that the subprogram Q has a transition probability matrix $\tilde{Q}$, where $\tilde{Q}$ specifies probabilities of transitions from B to $S = B \cup B^c$. Then the matrix associated with the whole program is easily seen to be

$$P = \left(\begin{array}{c|c} \tilde{Q} \\ \hline 0 \mid I \end{array}\right) = \begin{pmatrix} Q_1 & Q_2 \\ 0 & I \end{pmatrix}$$

where the states in B precede those in B^c, so that Q_1 is a transition (substochastic) matrix from B to B, and Q_2 is a similar matrix from B to B^c. A direct calculation shows that

$$P^n = \begin{pmatrix} Q_1^n & (Q_1^{n-1} + Q_1^{n-2} + \cdots + Q_1 + I)Q_2 \\ 0 & I \end{pmatrix}.$$

Hence, it follows from preceding remarks that if $i \in B$, $j \in B^c$ we have

$$f_{ij}^* = \sum_{k \geq 0} (Q_1^k Q_2)_{ij}.$$

But, in Kozen's notation, Q_1 is the matrix defining the linear operator $e_B \circ T_Q \circ e_B$ on the space of measures on S, whereas Q_2 is the matrix defining the operator $e_{B^c} \circ T_Q \circ e_B$, so that

$$f_{ij}^* = \left(\left[\left(\sum_{k \geqq 1} e_{B^c} \circ (T_Q \circ e_B)^k \right) (\delta_i) \right] (\{j\}) \right) = T(\delta_i)(\{j\})$$

where T is the distribution-transforming operator associated by Kozen with the while loop. This and (1) show that the two approaches indeed coincide for the above program.

Returning to the example program, it can be checked that

$$f^*_{(l_1,i),(l_2,j)} = p^{j-i}q \quad \text{if } j \geqq i,$$

and that

$$\mu_j^* = p^j q.$$

Unfortunately, in the general case, the quantities f_{ij}^* and μ_j^* may be difficult to compute explicitly, and we may be interested only in a partial property of the program. For example, we might only be interested in determining the expected number of steps till a 1 is chosen. This is the same as determining the expected value of i upon termination, which is

$$\sum_{i \geqq 0} i\mu^*_{(l_2,i)}.$$

We therefore would like to find methods for the calculation of such quantities without having to compute explicitly $\bar{\mu}^*$. This is done in the following sections.

2. Probabilistic verification by invariants. In this section we present our first probabilistic verification method. Motivated by the concluding remarks of the preceding section, we set out to find a way to compute a linear functional over $\bar{\mu}^*$, having the general form

$$\psi(\bar{\mu}^*) = \sum_{j \in T} \beta_j \mu_j^*.$$

This is a probabilistic analogue of an assertion on the terminal program states. We will assume, henceforth, that $\beta_j \geqq 0$, $j \in T$.

Our approach is to try to extend the coefficients $\{\beta_j\}_{j \in T}$ to a vector $\bar{\beta} = \{\beta_i\}_{i \in S}$ such that $\beta_i \geqq 0$, $i \in S$, and such that its restriction to T gives the original coefficients of ψ. Furthermore, we require $\bar{\beta}$ to be a right-characteristic vector of P, i.e., $P\bar{\beta} = \bar{\beta}$, or, in expanded form

$$\sum_{j \in S} P_{ij}\beta_j = \beta_i, \qquad i \in S.$$

(We assume also that all the infinite sums involved converge.) Such a $\bar{\beta}$ is known in Markov chain theory as a *P-regular* or *P-harmonic* function (cf. [RV], [KSK]). Note that, unless the vector $\{\beta_i\}_{i \in T}$ is bounded, some components β_i, for $i \in I$, may be $+\infty$.

If such a $\bar{\beta}$ exists, then define the linear functional $\varphi(\bar{\mu}) = \sum_{i \in S} \mu_i \beta_i$ which, given as an argument a distribution $\bar{\mu}$ over the program states, computes a real number (possibly ∞).

Then if $\varphi(\bar{\mu}^0)$ is finite, so is $\varphi(\bar{\mu}^{(n)})$, i.e. the value of the functional after step n, and we have the following invariance relation:

$$\varphi(\bar{\mu}^0) = \varphi(\bar{\mu}^{(n)}), \qquad n = 0, 1, \cdots.$$

This invariance is a consequence of the computation:

$$\varphi(\bar{\mu}^{(n)}) = (\bar{\mu}^{(n)} \cdot \bar{\beta}) = \bar{\mu}^0 P^n \bar{\beta} = (\bar{\mu}^0 \cdot \bar{\beta}) = \varphi(\bar{\mu}^0)$$

since $\bar{\beta}$ is a characteristic vector of P.

Obviously this gives a method for deriving invariance relations for general programs. We may now rewrite this invariance as:

$$\sum_{i \in I} \mu_i^{(n)} \beta_i + \sum_{i \in T} \mu_i^{(n)} \beta_i = \varphi(\bar{\mu}^0).$$

If we let now n go to ∞, the second term in the sum keeps increasing (because, for each $i \in T$, $\mu_i^{(n)}$ increases and $\beta_i \geqq 0$), and is bounded by $\varphi(\bar{\mu}^0)$ so that it must converge to a limit. Consequently, so must the first term, leading to:

$$R + \sum_{i \in T} \mu_i^* \beta_i = \varphi(\bar{\mu}^0)$$

where $R = \lim_{n \to \infty} R_n = \lim_{n \to \infty} \sum_{i \in I} \mu_i^{(n)} \beta_i \geqq 0$. Thus in the general case, we can conclude that

$$\psi(\bar{\mu}^*) \leqq \varphi(\bar{\mu}^0).$$

In the case where we can show that $R = 0$, we have an exact equality,

$$\psi(\bar{\mu}^*) = \varphi(\bar{\mu}^0).$$

Thus, this method allows us to compute the desired "output assertion" directly from the input distribution. Let us consider, for the simple example program above, the functional given by:

$$\beta_{(l_1,i)} = i + \frac{p}{q}, \qquad \beta_{(l_2,i)} = i$$

(i.e. a nonnegative functional that extends the desired functional ψ). The invariance verification condition $P\bar{\beta} = \bar{\beta}$ amounts in this case to

$$p\beta_{(l_1,i+1)} + q\beta_{(l_2,i)} = \beta_{(l_1,i)}, \quad \text{or} \quad p\left(i + 1 + \frac{p}{q}\right) + qi = i + \frac{p}{q},$$

which is easily verifiable, and

$$\beta_{(l_2,i)} = \beta_{(l_2,i)},$$

which is immediate.

Accepting the easily verifiable $\sum_i \mu_{(l_1,i)}^{(n)} (i + p/q) \to 0$ (since $\mu_{(l_1,i)}^{(n)} = \delta_{i,n} p^n$, $R_n = p^n(n + p/q) \to 0$), we therefore conclude:

$$\psi(\bar{\mu}^*) = \sum_{i=0}^{\infty} \mu_{(l_2,i)}^* i = \sum \mu_{(l_1,i)}^0 \left(i + \frac{p}{q}\right) = \frac{p}{q}.$$

This, of course, establishes that the expected value of i on termination of the program is p/q. (Here we have explicitly checked that $\lim_n R_n = 0$; in the sequel we will suggest several simple conditions that imply the vanishing of this limit.)

Summarizing the conditions for applicability of this method we have:

(V1) $$\sum_{i \in S} \mu_i^0 \beta_i < \infty,$$

(V2) $$\bar{\beta} \geqq 0, \qquad P\bar{\beta} = \bar{\beta}.$$

Then, under these two conditions we are assured of

$$\psi(\bar{\mu}^*) = \sum_{i \in T} \mu_i^* \beta_i \leqq \sum_{i \in S} \mu_i^0 \beta_i = \varphi(\bar{\mu}^0).$$

In fact, in order to ensure this result it is sufficient to have $P\bar{\beta} \leqq \bar{\beta}$ in (V2).

If in addition we also have

(V3) $$\sum_{i \in I} \mu_i^{(n)} \beta_i \underset{n \to \infty}{\to} 0 \qquad (\beta\text{-termination}),$$

then we may conclude that

(C) $$\psi(\bar{\mu}^*) = \sum_{i \in T} \mu_i^* \beta_i = \sum_{i \in S} \mu_i^0 \beta_i = \varphi(\bar{\mu}^0).$$

In order to emphasize the similarity between this method and the method of intermediate assertions [FL], we point out that (V1) is analogous to saying that φ is true initially, while (V2) is analogous to the local verification conditions. Thus (V1) and (V2) imply (V3)$\Rightarrow$(C), but this is the analogue of partial correctness. It states that if the program converges then the value on convergence is equal to $\varphi(\bar{\mu}^0)$. Only here, we have to require an appropriate rate of convergence as well (i.e. β-termination).

Note that if the β_i, $i \in T$, are uniformly bounded, then (V3) will hold if the program terminates with probability 1. Indeed, then one has

$$\lim_n \sum_{i \in I} \mu_i^{(n)} \beta_i \leqq \sup_{i \in I} |\beta_i| \cdot \lim_{n \to \infty} \sum_{i \in I} \mu_i^{(n)},$$

but the right-hand-side limit is precisely the probability of the program not to terminate, which, by assumption, is 0. Similar sufficient conditions for the β-termination of the program can be given for other kinds of vectors $\bar{\beta}$ (see § 3 where such a general condition is given).

The main question concerning this approach is: Can we always extend a given functional ψ to an invariant functional φ in the manner described above (in other words, is this method complete)? To see that this is indeed the case, we proceed as follows: Let $\{\beta_j\}_{j \in T}$ be given, with $\beta_j \geqq 0$ for each $j \in T$. For each $i \in I$ define

(2) $$\beta_i = \sum_{j \in T} \beta_j f_{ij}^*.$$

Each sum exists in an extended sense (β_i can be $+\infty$). Let $\tilde{\psi}$ be the functional $\tilde{\psi}(\bar{\mu}) = \sum_{i \in S} \beta_i \mu_i$.

THEOREM 1. *If $\tilde{\psi}$ satisfies* (V1), *then it also satisfies* (V2) *and* (V3). *On the other hand, if $\tilde{\psi}$ does not satisfy* (V1), *then* $\psi(\bar{\mu}^*) = +\infty$.

Proof. We first note that (V2) always holds. Clearly $\beta_i \geqq 0$ for each $i \in I$, and

$$\sum_{k \in S} P_{ik}\beta_k = \sum_{k \in S} P_{ik} \sum_{j \in T} f^*_{kj}\beta_j.$$

Interchanging the order of summation, we obtain

$$= \sum_{j \in T} \beta_j \left(\sum_{k \in S} P_{ik} f^*_{kj} \right).$$

But by the monotone convergence theorem,

$$\sum_{k \in S} P_{ik} f^*_{kj} = \sum_{k \in S} P_{ik} \lim_{n \to \infty} P^n_{kj} = \lim_{n \to \infty} \sum_{k \in S} P_{ik} P^n_{kj} = \lim_{n \to \infty} P^{n+1}_{ij} = f^*_{ij}.$$

Hence

$$\sum_{k \in S} P_{ik}\beta_k = \sum_{j \in T} \beta_j f^*_{ij} = \beta_i.$$

For $i \in T$, $P_{ik} = \delta_{ik}$, so certainly $\sum_{k \in S} P_{ik}\beta_k = \beta_i$. (Equality also holds when both sides are $+\infty$.) Hence (V2) is satisfied.

Next, suppose that (V1) holds for $\tilde{\psi}$. Again, by substituting the value of β_i, $i \in I$, and interchanging the order of summation, we obtain

$$\sum_{i \in S} \mu^0_i \beta_i = \sum_{j \in T} \beta_j \left(\sum_{l \in S} \mu^0_i f^*_{ij} \right) < \infty.$$

That is,

$$\sum_{i \in S} \mu^0_i \beta_i = \sum_{j \in T} \beta_j \mu^*_j < \infty.$$

This already shows that (C) holds, but it also follows from this that

$$R = \lim_{n \to \infty} \sum_{i \in I} \mu^{(n)}_i \beta_i = \tilde{\psi}(\bar{\mu}^0) - \psi(\bar{\mu}^*) = 0.$$

Hence (V3) also holds. If (V1) does not hold, we still have the above equality $\tilde{\psi}(\bar{\mu}^0) = \psi(\bar{\mu}^*) = +\infty$. Q.E.D.

This, of course, establishes the completeness of our verification method theoretically. That is, given a partial vector $\beta_j \geqq 0$, $j \in T$, there always exists a completion of it to a full vector β_i, $i \in S$, which satisfies (V2) and satisfies either (V1), (V3) and (C), or else the value $\psi(\bar{\mu}^*) = \sum_{j \in T} \beta_j \bar{\mu}^*_j = +\infty$.

In practice, of course, there are some difficulties. First, in order to obtain the above completion of $\bar{\beta}$, we need to know the matrix f^*_{ij}, which is generally unavailable. Similarly, the establishment of (V3) (for any completion of $\bar{\beta}$) requires the knowledge of $\bar{\mu}^{(n)}$ for every $n \geqq 0$ which is also unavailable.

A partial solution to these problems is given by the following characterization of the specific completion of $\bar{\beta}$ given by (2).

PROPOSITION 1. *Let $\beta_j \geqq 0$, $j \in T$, be given. Then the completion of $\bar{\beta}$ given by* (2) *is the smallest nonnegative P-regular (i.e. invariant) extension of the given β_j's.*

Proof. Let $\bar{\gamma} = \{\gamma_i\}_{i \in I}$ be such that $\gamma_i \geqq 0$ for all $i \in I$, and such that, together with the given $\bar{\beta}^0 \equiv \{\beta_j\}_{j \in T}$, $\bar{\gamma}$ is P-regular. Let us decompose P into blocks as follows:

$$P = \begin{matrix} \\ I \\ T \end{matrix} \overset{\begin{matrix} I & T \end{matrix}}{\begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}} \begin{matrix} \} I \\ \} T \end{matrix}.$$

The invariance of γ then can be written as

$$Q\bar{\gamma} + R\bar{\beta}^0 = \bar{\gamma}.$$

Since $\bar{\gamma} \geqq 0$, we have $Q\bar{\gamma} \geqq 0$ and hence $\bar{\gamma} \geqq R\bar{\beta}^0$. Continuing inductively in this manner, we obtain

$$\bar{\gamma} \geqq \sum_{k \geqq 0} Q^k R \bar{\beta}^0.$$

A computation completely analogous to the one performed in the preceding section yields

$$\left(\sum_{k \geqq 0} Q^k R \bar{\beta}^0 \right)_i = \lim_{n \to \infty} \left(\sum_{j \in T} P_{ij}^{(n)} \beta_j \right) = \sum_{j \in T} f_{ij}^* \beta_j = \beta_i.$$

(Interchanging the limit and the sum is justified by the monotone convergence theorem.) Hence $\gamma_i \geqq \beta_i$ for each $i \in I$. Q.E.D.

Proposition 1 therefore yields a practical method for probabilistic verification. Starting with the given coefficients $\beta_j \geqq 0$, $j \in T$, we find the general solution to the invariance equations $P\bar{\beta} = \bar{\beta}$, that coincides with the given partial vector on T, and then choose the smallest nonnegative solution from which the value of $\psi(\bar{\mu}^*)$ can be readily obtained.

As an illustration, consider our example program with the coefficients $\beta_{(l_2,i)} = i$.

Example 1. The recurrence equations implied by $P\bar{\beta} = \bar{\beta}$ are

$$p\beta_{(l_1,i+1)} + qi = \beta_{(l_1,i)},$$

whose general solution is readily found to be

$$\beta_{(l_1,i)} = i + \frac{p}{q} + \frac{A}{p^i}.$$

By the requirement of minimality (and nonnegativity) we must choose $A = 0$ and are ensured, by the above two propositions, that conditions (V1)–(V3) are satisfied, so that calculation of $\psi(\bar{\mu}^*)$ can proceed as before.

Thus, we may summarize: In order to compute $\sum_{j \in T} \mu_j^* \beta_j$, find a completion β_i, $i \in I$, such that (V1) and (V2) are satisfied and either:

a) β_i, $i \in I$, is the minimal such solution; or
b) The program terminates with probability 1 and the β_i, $i \in I$, are uniformly bounded; or
c) It is possible to verify (V3) by some other means.

(More about such means will be discussed in the next section.) The desired value is then $\sum_{i\in S}\beta_i\mu_i^0$. In case a), if this latter value is $+\infty$, then so is the desired value of the "output assertion". Note that the family of nonnegative linear functionals enables us to express a very rich variety of program properties upon termination, such as:

(i) The probability of terminating at a particular state $j\in T$ (take $\beta_k=\delta_{kj}$).
(ii) The probability of termination (take $\beta_j\equiv 1$).
(iii) The expected value of some variable (as in our running example).
(iv) The expected running time of the program. (One way of doing this is to add a special "step-counter" variable to the program, and then find its expected value (cf. [RA]).)
(v) Higher moments of some variables, such as variance, etc.

Remark 1. If one knows that the program terminates almost surely (i.e. with probability 1), then one can generalize the second approach mentioned above to the case where the β_i's are not bounded. This is simply done by defining a sequence of partial vectors

$$\beta_j^{(N)}=\min(\beta_j, N),\quad j\in T,\quad N=1,2,\cdots.$$

For each N, $\beta_j^{(N)}\leqq N$, $j\in T$. Hence, the smallest completion of the $\beta_j^{(N)}$'s, given by (2), is also uniformly bounded (by N).

We can therefore consider any bounded invariant completion $\bar\beta^{(N)}$ of the coefficients $\beta_j^{(N)}$, $j\in T$, and obtain

$$\psi^{(N)}(\bar\mu^*)=\sum_{j\in T}\mu_j^*\beta_j^{(N)}=\sum_{i\in S}\mu_i^0\beta_i^{(N)}=\varphi^{(N)}(\mu^0).$$

By the monotone convergence theorem, $\lim_{N\to\infty}\psi^{(N)}(\bar\mu^*)=\sum_{j\in T}\mu_j^*\beta_j$. Hence, this sum is also equal to $\lim_{N\to\infty}\varphi^{(N)}(\bar\mu^0)$.

Remark 2. The techniques used in this section are rather standard in Markov chain theory (cf. [RV], [KSK]). It is pleasing to find out that the adaptation of Markov chain theory to the realm of program verification yields a natural and straightforward generalization of existing verification methods for deterministic programs.

We conclude this section with two additional illustrations of our method

Example 2. Consider the program

$x:=0$;
l_1: **while** $(t:=\text{random}(\frac{1}{3}\delta_0+\frac{1}{3}\delta_1+\frac{1}{3}\delta_2))\neq 2$ **do** $x:=x+t$;
l_2: **halt**.

We associate states with the location in the program and with the value of x. Hence, we can write

$$I=\{(l_1,n)|n\geqq 0\},\qquad T=\{(l_2,n)|n\geqq 0\}.$$

The nonzero transition probabilities for this program are

$$P_{(l_1,n),(l_1,n)}=\tfrac{1}{3},\quad P_{(l_1,n),(l_1,n+1)}=\tfrac{1}{3},\quad P_{(l_1,n),(l_2,n)}=\tfrac{1}{3},\quad P_{(l_2,n),(l_2,n)}=1.$$

Let us compute the terminal distribution $\bar\mu^*$. We thus fix $n\geqq 0$, and put

$$\beta_{(l_2,j)}=\delta_{j,n}.$$

We then want to extend $\bar\beta$ over the nonterminal state as well, so that it is invariant. The requirement $P\bar\beta=\bar\beta$ then becomes

$$\tfrac{1}{3}\beta_{(l_1,i)}+\tfrac{1}{3}\beta_{(l_1,i+1)}+\tfrac{1}{3}\delta_{i,n}=\beta_{(l_1,i)},\quad i\geqq 0,\quad\text{or}\quad \beta_{(l_1,i+1)}=2\beta_{(l_1,i)}-\delta_{i,n},$$

whose general solution is

$$\beta_{(l_i,i)} = \begin{cases} K \cdot 2^i, & i \leqq n, \\ K \cdot 2^i - 2^{i-n-1}, & i > n. \end{cases}$$

To obtain the smallest nonnegative solution, we must choose $K = 2^{-n-1}$. Hence, we conclude that

$$\mu^*_{(l_2,n)} = \sum_{i=0}^{n} 2^{i-n-1} \mu^0_i.$$

In particular, since $\mu^0_{(l_1,i)} = \delta_{i,0}$, we obtain $\mu^*_{(l_2,n)} = 1/2^{n+1}$. (Note that in this example we have actually computed the matrix f^*_{ij}, in a somewhat roundabout way.)

Example 3 (Gambler's ruin or Drunkard's walk). Consider the following program

$x := n$; /* n is some positive integer */
l_1: **while** $x \neq 0$ **do** $x := x + \text{random}\,(p\delta_{-1} + q\delta_1)$;
l_2: **halt**.

This program simulates a random walk on the nonnegative integers with 0 as an "absorbing barrier." This describes a process in which a gambler with an initial fortune n plays indefinitely against a house with unlimited fortune. In each game the player has a chance p of losing and a chance $q = 1 - p$ of winning. The process stops when the gambler loses all its money.

States are defined as in the preceding example, but in this case T contains only the state $(l_2, 0)$. The nonzero transition probabilities are

$$P_{(l_1,j),(l_1,j-1)} = p, \qquad P_{(l_1,j),(l_1,j+1)} = q \quad \text{for } j > 0,$$
$$P_{(l_1,0),(l_2,0)} = P_{(l_2,0),(l_2,0)} = 1,$$

and the initial distribution is $\mu^0_{(l_1,j)} = \delta_{j,n}$. Let us compute the probability of the program termination, i.e. taking $\beta_{(l_2,0)} = 1$. Extending $\bar{\beta}$ as usual, we are led to the following recurrence equations:

$$\beta_{(l_1,0)} = 1, \qquad \beta_{(l_1,j)} = p\beta_{(l_1,j-1)} + q\beta_{(l_1,j+1)}, \qquad j > 0.$$

Solution of these recurrence equations amounts to solving the equation

$$q\lambda^2 - \lambda + p = 0$$

whose roots are 1 and p/q. If $p \neq \frac{1}{2}$, the roots are distinct and the general solution is

$$\beta_{(l_1,j)} = 1 + K\left[\left(\frac{p}{q}\right)^j - 1\right].$$

If $p > \frac{1}{2}$, then $p/q > 1$ and the minimality requirement forces us to choose $K = 0$. Hence, $\beta_{(l_1,j)} = 1$ for all $j \geqq 0$ so that the termination probability is $\beta_{(l_1,n)} = 1$ (the gambler will almost surely be ruined).

If $p < \frac{1}{2}$, then $p/q < 1$, and we choose

$$K = \inf \frac{1}{1 - (p/q)^j} = 1.$$

Hence, $\beta_{(l_1,j)} = (p/q)^j$, so that the termination probability is $(p/q)^n$.

Finally, if $p = \frac{1}{2}$, the λ-equation has two identical roots, and the general solution for the β's is

$$\beta_{(l_1,j)} = 1 + Kj.$$

The minimality condition implies $K = 0$ so that $\beta_{(l_1,n)} = 1$ and the program terminates almost surely in this case.

As a related example, consider the case $p = \frac{1}{2}$ in the above program. Although the program terminates almost surely in this case, it is well known [CH] that it is not expected to terminate. We will establish this fact using our method. To do so, we need to introduce an additional step-counter variable c, and modify the program as follows:

$[x, c] := [n, 0]$;
l_1: **while** $x \neq 0$ **do** $[x, c] := [x + \text{random}(\frac{1}{2}\delta_{-1} + \frac{1}{2}\delta_1), c+1]$;
l_2: **halt**.

The program states are now

$$I = \{(l_1, c, n) | c \geqq 0, n \geqq 0\}, \qquad T = \{(l_2, c, 0) | c \geqq 0\}.$$

The nonzero transition probabilities are

$$P_{(l_1,c,j),(l_1,c+1,j+1)} = P_{(l_1,c,j),(l_1,c+1,j-1)} = \tfrac{1}{2}, \qquad j > 0$$
$$P_{(l_1,c,0),(l_2,c,0)} = P_{(l_2,c,0),(l_2,c,0)} = 1.$$

The initial distribution is $\mu^0_{(l_1,0,n)} = 1$, and zero elsewhere. We wish to compute the expected value of c upon termination, so that we begin with the partial vector $\beta_{(l_2,c,0)} = c$, $c \geqq 0$, and we wish to extend it to an invariant vector $\bar{\beta}$. Instead of doing so directly, we use the limiting approach suggested in Remark 1 above. That is, let $M \geqq 0$ be an integer, and define

$$\beta^M_{(l_2,c,0)} = \begin{cases} c, & c \leqq M, \\ 0, & c > M. \end{cases}$$

Since β^M is uniformly bounded on T, and the program is already known to terminate almost surely, *any* invariant completion $\bar{\beta}^M$ can be used to compute the desired functional. We claim that the following is such an invariant completion:

$$\beta^M_{(l_1,c,j)} = \sum_{k=0}^{\lfloor (M-c-j)/2 \rfloor} \frac{c+j+2k}{2^{j+2k}} \left[\binom{j+2k-1}{k} - \binom{j+2k-1}{k-1} \right], \qquad j \geqq 1,$$

(note that the sum vanishes if $c + j > M$), and

$$\beta^M_{(l_1,c,0)} = \begin{cases} c, & c \leqq M, \\ 0, & c > M. \end{cases}$$

To verify that $\bar{\beta}^M$ has the desired properties, one has to check that the following equations hold:

$$\beta^M_{(l_1,c,0)} = \beta^M_{(l_2,c,0)} \quad \text{and} \quad \beta^M_{(l_1,c,j)} = \tfrac{1}{2}\beta^M_{(l_1,c+1,j+1)} + \tfrac{1}{2}\beta^M_{(l_1,c+1,j-1)}, \qquad j \geqq 1.$$

The first equation is immediate, and the second can easily be checked. Since β^M is uniformly bounded on I (it is zero on all but a finite number of components), we conclude that

$$E^M(c) = \sum_{c=0}^{M} c\mu^*_{(l_2,c,0)} = \beta^M_{(l_1,0,n)} = \sum_{k=0}^{\lfloor (M-n)/2 \rfloor} \frac{n+2k}{2^{n+2k}} \left[\binom{n+2k-1}{k} - \binom{n+2k-1}{k-1} \right].$$

Hence, if we let $M \to \infty$, we find that the expected value of c is the sum of the infinite series appearing above. Using Stirling's formula, we find that the kth term of this series is of the order of $k^{-1/2}$, so that the series diverges and the expected value of c is infinite. (This method can be used to show that the αth moment of c is finite for $\alpha < \frac{1}{2}$ and infinite for $\alpha > \frac{1}{2}$.)

3. A dual approach—expected number of visits. A recent work of Ramshaw [RA] suggests an alternative approach to probabilistic program verification. Although he does not use Markov chains in his approach, it turns out that his approach can be easily and naturally described in terms of the Markov chain model that we have been using. This leads to a much more compact description of his method, helps to explain the problems that it faces and the (partial) solutions to these problems suggested by Ramshaw, and also makes it easier to generalize this approach and to connect it with our first approach as given in the preceding section. All this will be done in this section.

Intuitively, the approach that we have taken in the first section was to record the program behavior by taking "snapshots" of the distribution of all program states, at different times during execution. An invariant functional is thus a linear "assertion" about this distribution that does not change from one snapshot to another. The approach that Ramshaw takes is orthogonal to ours, in the sense that he takes an "infinite-exposure" picture, of each program state separately, throughout the program execution. His approach can be formally explained in terms of the Markov chain model as follows:

Let P be the transition probability of the program. Decomposing it into blocks as we did in the previous section, we obtain

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \begin{matrix} \}I \\ \}T \end{matrix}$$
$$\underbrace{}_{I} \quad \underbrace{}_{T}$$

(i.e., Q_{ij} is the probability of going from $i \in I$ to $j \in I$, and R_{ij} is the probability of going from $i \in I$ to $j \in T$). Consider a modified transition matrix defined as

$$\tilde{P} = \begin{pmatrix} Q & R \\ 0 & 0 \end{pmatrix}.$$

This matrix corresponds to a (substochastic) process in which, once the process reaches a terminal state, it stops right there.

Let $\tilde{\mu}^{(n)}$ be the distribution of program states after n steps of the revised process, i.e. $\tilde{\mu}^{(n)} = \bar{\mu}^0 \tilde{P}^n$. Define a vector $\bar{v}$ over S as follows:

$$\bar{v} = \sum_{n=0}^{\infty} \tilde{\mu}^{(n)}. \tag{3}$$

For each $i \in S$, v_i is the expected number of visits at state i in an execution, given the initial distribution $\bar{\mu}^0$. Following Markov chain terminology, $\bar{v}$ is a pure potential measure induced by the charge $\bar{\mu}^0$. Note that $\bar{v}$ is always defined in an extended sense, but need not be finite. However, if $i \in T$, v_i is always finite and in fact we have $v_i = \mu_i^*$. This follows from the fact that terminal states are visited at most once in the revised process.

From the definition of $\bar{v}$ we have immediately the following

CLAIM. *$\bar{v}$ is the smallest nonnegative solution of the equation*

$$\bar{v} = \bar{v}\tilde{P} + \bar{\mu}^0. \tag{4}$$

Ramshaw's approach is to consider the quantities v_i, and to introduce assertions about them having the following restricted form:

$$\sum_{i \in A} v_i = e$$

where $A \subseteq S$ is a subset of states all having the same program location (he refers to these assertions as "vanilla" assertions). His method is to verify that these assertions are consistent with (4), i.e., to show that any vector $\bar{v}$ satisfying (4) also satisfies the assertions. This is done by a generalization of the standard inductive assertions method, but may not always work. In fact, the assertions must be of a special structure to allow his inference rules to be applicable.

Ramshaw shows that under certain conditions (roughly amounting to requiring that $\bar{v}$ be finite) this proof method is sound and yields some information about $\bar{\mu}^*$, given by those assertions that are planted at the program termination point. Ramshaw does not bother to actually solve (4), and so the main problem that he faces is to show that his assertions, even when consistent with (4), do actually describe the smallest solution of (4). This creates the possibility of obtaining nonminimal solutions, such as his so-called "time bombs", which may not yield the desired $\bar{\mu}^*$.

Having stated the basic nature of Ramshaw's approach, we will not follow his method of estimating $\bar{v}$. Rather, we view the solution of (4) as the main goal of this approach. In this regard, there are several additional possibilities for estimating $\bar{v}$. For example,

A) Let $\bar{u}$ be any nonnegative solution of

$$\bar{u} = \bar{u}\tilde{P} + \bar{\mu}^0$$

or even of

$$\bar{u} \geqq \bar{u}\tilde{P} + \bar{\mu}^0.$$

Then for every i, $v_i \leqq u_i$, so that $\bar{u}$ is an approximation from above to the desired $\bar{v}$.

B) Define the sequence of vectors

$$\bar{v}^0 = \bar{\mu}^0, \qquad \bar{v}^{n+1} = \bar{v}^n\tilde{P} + \bar{\mu}^0, \qquad n \geqq 0.$$

Then for every i and $n \geqq 0$, $v_i^n \leqq v_i$, so that $\bar{v}^n$ is an approximation from below for $\bar{v}$.

To illustrate this approach, let us return to our running example of the program that searches for a first appearance of 1 in an infinite sequence of independent draws of 0 and 1. Equation (4) then has the following form:

$$\begin{aligned} v_{(l_1,i)} &= \mu^0_{(l_1,i)} + pv_{(l_1,i-1)}, \qquad i \geqq 1, \\ v_{(l_1,0)} &= \mu^0_{(l_1,0)}, \\ v_{(l_2,i)} &= qv_{(l_1,i)}, \qquad i \geqq 0, \end{aligned}$$

and the (unique) solution is easily found to be

$$v_{(l_1,i)} = p^i, \qquad v_{(l_2,i)} = p^i q = \mu^*_{(l_2,i)}, \qquad i \geqq 0.$$

Drawing an analogy to the standard verification techniques, we can compare this second approach to the computational induction method [PA], where information about intermediate program states is derived inductively from information about its input states. Comparing the two methods presented in this paper, we may view the first one as being *goal-directed*, in that it draws from the program output requirements conditions that should hold at the intermediate and input program states, and only then checks them against the input information about the program. On the other hand, the second method is *input-directed*, in that it draws information about intermediate and terminal program states from the input information and then computes the output data from this information.

There are two main disadvantages of the second approach. The first is that it is, as just pointed out, input-dependent. Hence, if the program input distribution is not fixed then we may have to recompute the vector $\bar{v}$ afresh for each new distribution $\bar{\mu}^0$. By contrast, it is more natural to assume that the questions about the program terminal states are fixed, and we can process each of them using our first method independently of any input distribution. Then for each input distribution, we can compute the required output quantities immediately.

A second disadvantage is that the expected number of visits at an intermediate state need not be finite even if, say, the program terminates almost surely. Hence in solving (4), we may find that certain v_i's are $+\infty$. This is not a major obstacle, since (4) holds even in such a case. This means that each finite component v_i, $i \in I$, cannot depend on any infinite components, so that these infinite components correspond to states from which the program almost surely diverges and cannot reach a terminal state with a positive probability.

Aside from independent computation of the $\bar{v}$ or approximations thereof, we can connect solutions to (4) to linear invariant functionals. Let $\bar{u}$ be any solution to $\bar{u} \geqq \bar{u}\tilde{P} + \bar{\mu}^0$ and $\bar{\beta}$ any solution to $P\bar{\beta} = \bar{\beta}$. By partitioning $\bar{u} = (\bar{u}_1, \bar{u}_2)$ and $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2)$ according to the partitioning of S into I, T, we obtain the following equations satisfied by each:

$$\bar{u}_1 \geqq \bar{u}_1 Q + \bar{\mu}_1^0, \quad \bar{u}_2 \geqq \bar{u}_1 R + \bar{\mu}_2^0, \quad \bar{\beta}_1 = Q\bar{\beta}_1 + R\bar{\beta}_2.$$

Obviously $\bar{\mu}_1^{(n)} = \bar{\mu}_1^0 Q^n$. Thus $\bar{v}_1 = \sum_{n=0}^{\infty} \bar{\mu}_1^{(n)}$.

Consequently $(\bar{u}_1 \cdot \bar{\beta}_1) \geqq (\bar{v}_1 \cdot \bar{\beta}_1) = \sum_{n=0}^{\infty} (\bar{\mu}_1^{(n)} \cdot \bar{\beta}_1) = \sum_{n=0}^{\infty} R_n$ where R_n is the remainder such that $\lim_{n\to\infty} R_n = 0$ is the required (V3) condition. Hence, we have

PROPOSITION 2. *If for some* $\bar{u}_1$ *satisfying*

$$\bar{u}_1 \geqq \bar{u}_1 Q + \bar{\mu}_1^0$$

and some $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2)$ *satisfying*

$$\bar{\beta}_1 = Q\bar{\beta}_1 + R\bar{\beta}_2$$

the product $(\bar{u}_1 \cdot \bar{\beta}_1) < \infty$, *then* (C) *holds*; *i.e.*,

$$\bar{\mu}_2 \cdot \bar{\beta}_2 = \bar{\mu}_1^0 \cdot \bar{\beta}_1 + \bar{\mu}_2^0 \cdot \bar{\beta}_2.$$

Note, however, that this is only a sufficient condition which implies the $\bar{\beta}$-termination of the program (i.e. condition (V3)), but is not necessarily equivalent to it. To illustrate this point, suppose that $\bar{\beta} \equiv 1$. This is an invariant vector whose restriction to T yields a functional that computes the probability of termination. If $\bar{\beta}$ satisfies (V1)–(V3), and therefore also (C), then it follows that the program almost surely terminates. On the other hand, the above condition, even for $\bar{u} = \bar{v}$, reads

$$\sum_{i \in I} v_i < \infty.$$

However, we have

LEMMA 1. $\sum_{i \in I} v_i < \infty$ *if and only if the program has a finite expectation of termination* (*i.e., the expected length of stay in I is finite*), *which is then equal to that sum.*

Proof. The expectation of program termination is

$$\sum_{n\geqq 0} n \cdot \text{Prob (the program terminates in exactly } n \text{ steps)}$$

$$= \sum_{n\geqq 1} \text{Prob (the program terminates only after } n \text{ or more steps)}$$

$$\leqq \sum_{n\geqq 0} \text{Prob (the program is not yet in } T \text{ after } n \text{ steps)}$$

$$= \sum_{n\geqq 0} \sum_{i\in I} \tilde{\mu}_i^{(n)} = \sum_{i\in I} v_i.$$

Hence $\sum_{i\in I} v_i < \infty$ implies a finite expectation of termination.

Conversely, if the program has a finite expectation to terminate, then it terminates almost surely, which makes the inequality in the above formulae into an equality. hence $\sum_{i\in I} v_i$ is equal to the expectation to terminate, which is finite. Q.E.D.

Hence the above condition requires that the program have a finite expected execution length, which in general is stronger than the requirement that it terminate almost surely.

Nevertheless, we have the following alternative approach to verification:

(i) Find any nonnegative solution $\bar{u}$ of (4) (even with an inequality).

(ii) Find an invariant nonnegative completion $\bar{\beta}$ of the coefficients of the given functional on T.

(iii) Check that $(\bar{u}_1 \cdot \bar{\beta}_1) < \infty$, where $\bar{\beta}_1 = \bar{\beta}|_I$, $\bar{u}_1 = \bar{u}|_I$.

If so, $\bar{\beta}$-termination, and hence also (C), are assured.

An example of this procedure will be given in the following section. We will conclude this section with an example of a straightforward calculation of $\bar{v}$. Consider the Gambler's Ruin program given in Example 3, with $p = q = \frac{1}{2}$. Following the notation of § 2, equations (4) have the form

$$\begin{aligned}
v_{(l_1,0)} &= \tfrac{1}{2} v_{(l_1,1)}, \\
v_{(l_1,1)} &= \tfrac{1}{2} v_{(l_1,2)}, \\
v_{(l_1,j)} &= \tfrac{1}{2} v_{(l_1,j-1)} + \tfrac{1}{2} v_{(l_1,j+1)}, \qquad j \neq n, \quad j > 1, \\
v_{(l_1,n)} &= \tfrac{1}{2} v_{(l_1,n-1)} + \tfrac{1}{2} v_{(l_1,n+1)} + 1, \\
v_{(l_2,0)} &= v_{(l_1,0)}.
\end{aligned}$$

Putting $v_{(l_1,0)} = a$, we have the following general solution:

$$v_{(l_1,j)} = \begin{cases} a, & j = 0, \\ 2aj, & j = 1, \cdots, n, \\ 2aj - 2(j-n), & j > n. \end{cases}$$

Since we want the smallest nonnegative solution of (4), we must take $a = 1$, and we obtain the solution

$$v_{(l_1,j)} = \begin{cases} 1, & j = 0, \\ 2j, & j = 1, \cdots, n, \\ 2n, & j > n. \end{cases}$$

Hence $\mu^*_{(l_2,0)} = v_{(l_2,0)} = v_{(l_1,0)} = 1$. This means that the program terminates almost surely. However, $\sum_{i\in I} v_i = +\infty$, so that the program is not expected to terminate, in accordance with the results obtained, in a much more complicated manner, in the preceding section.

4. Additional examples. In this section we will illustrate the verification methods developed in the two preceding sections, as applied to two nontrivial example programs.

Example 4. Consider the following program ($0<\alpha<1$ is fixed):

$y := 0$; $n := 1$;
while $y < \alpha$ **do**
  $y := y + 1/2^n$ random $(\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1)$;
  $n := n+1$;
od.

For simplicity, we identify states only by the values of y and n. Thus, $S = \{(y, n)\colon n \geqq 1, y \in D_{n-1}\}$, where D_j is the set of all dyadic fractions having j binary digits. The terminating states are

$$T = \{(y, n)\colon n \geqq 1,\ y \in D_{n-1},\ y > \alpha \text{ and the } (n-1)\text{st digit of } y \text{ is } 1\},$$

and the transition probabilities are

$$\left.\begin{aligned} &P_{(y,n),(y,n+1)} = \tfrac{1}{2} \\ &P_{(y,n),(y+1/2^n,n+1)} = \tfrac{1}{2} \end{aligned}\right\} \quad y \in D_{n-1},\quad y \leqq \alpha \quad \text{(i.e. } (y, n) \in I\text{)},$$

$$P_{(y,n),(y,n)} = 1 \quad y \in D_{n-1},\quad y > \alpha \quad \text{(i.e. } (y, n) \in T\text{)}.$$

Let us compute the termination probability of this program; that is, we wish to compute

$$\psi(\bar{\mu}^*) = \sum_{(y,n)\in T} \mu^*_{(y,n)}.$$

Let us extend ψ to an invariant nonnegative functional φ:

$$\varphi(\bar{\mu}) = \sum_{(y,n)\in S} \beta_{(y,n)}\mu_{(y,n)}, \quad \text{where } \beta_{(y,n)} = 1 \quad \text{for } (y, n) \in T.$$

The invariance of φ implies that for each $(y, n) \in I$ we must have

$$\beta_{(y,n)} = \tfrac{1}{2}\beta_{(y,n+1)} + \tfrac{1}{2}\beta_{(y+(1/2^n),n+1)}.$$

Of course, these equations have the solution $\beta_{(y,n)} \equiv 1$, but this is too large. (Note in general that such an extension implies that $\varphi(\bar{\mu}^0) = 1$ for any initial distribution $\bar{\mu}^0$. Hence, this extension is the smallest nonnegative invariant extension of ψ if and only if the program terminates almost surely, regardless of the initial distribution.) In our case, we have a smaller solution:

$$\beta_{(y,n)} = \begin{cases} 0, & 2^{n-1}(\alpha - y) > 1, \\ 1, & 2^{n-1}(\alpha - y) < 0, \\ 1 - 2^{n-1}(\alpha - y), & \text{otherwise.} \end{cases}$$

Indeed, let us verify that these coefficients satisfy the above recurrence equation. Suppose first that $\beta_{(y,n)} = 1$. This happens when $y \geqq \alpha$, and then both $\beta_{(y,n+1)} = \beta_{(y+1/2^n,n+1)} = 1$, so the equation is satisfied. Next suppose $\beta_{(y,n)} = 0$. This happens when $y \leqq \alpha - 1/2^{n-1}$. Then obviously $y \leqq \alpha - 1/2^n$ so that $\beta_{(y,n+1)} = 0$, and also

$y+1/2^n \leqq \alpha - 1/2^n$, so that $\beta_{(y+1/2^n,n+1)} = 0$. Hence the equation is satisfied in this case, too. (Note that this corresponds to the case where the program never terminates if it reaches the state (y, n).) Finally, assume that

$$0 < 2^{n-1}(\alpha - y) < 1, \quad \text{i.e.,} \quad \alpha - \frac{1}{2^{n-1}} < y < \alpha.$$

Two subcases are possible. If $\alpha - 1/2^n \leqq y < \alpha$ then $y + 1/2^n \geqq \alpha$ and so $\beta_{(y+1/2^n,n+1)} = 1$, whereas $\beta_{(y,n+1)} = 1 - 2^n(\alpha - y)$. Hence

$$\tfrac{1}{2}\beta_{(y+1/2^n,n+1)} + \tfrac{1}{2}\beta_{(y,n+1)} = \tfrac{1}{2} + \tfrac{1}{2} - 2^{n-1}(\alpha - y) = 1 - 2^{n-1}(\alpha - y) = \beta_{(y,n)}.$$

If $\alpha - 1/2^{n-1} < y < \alpha - 1/2^n$, then $\alpha - 1/2^n < y + 1/2^n < \alpha$, so that

$$\beta_{(y+1/2^n,n+1)} = 1 - 2^n\left(\alpha - y - \frac{1}{2^n}\right),$$

whereas $\beta_{(y,n+1)} = 0$. Hence

$$\frac{1}{2}\beta_{(y,n+1)} + \frac{1}{2}\beta_{(y+1/2^n,n+1)} = 0 + \frac{1}{2} - 2^{n-1}\left(\alpha - y - \frac{1}{2^n}\right) = 1 - 2^{n-1}(\alpha - y) = \beta_{(y,n)}.$$

Note that we have not shown that these $\beta_{(y,n)}$'s yield the smallest invariant extension of ψ (although this is indeed the case). However, we can apply the duality principle stated in § 3 by computing the vector $\bar{v}_1$ and checking that $(\bar{v}_1 \cdot \bar{\beta}_1) < \infty$. As it turns out, computation of $\bar{v}_1$ in this case is simpler, because each nonterminating state (y, n) can be reached from only one preceding state. Specifically, we have

$$v_{(0,1)} = 1, \qquad v_{(y,n)} = \tfrac{1}{2}v_{(y',n-1)}, \qquad n > 1, \quad y \in D_{n-1}, \quad y \leqq \alpha,$$

where $y' \in D_{n-2}$ consists of the first $n-2$ digits of y. Thus, for each $(y, n) \in I$, we have

$$v_{(y,n)} = \frac{1}{2^{n-1}}.$$

Now we have

$$\sum_{(y,n)\in I} v_{(y,n)}\beta_{(y,n)} < \infty$$

because for each n there are at most two $y \in D_{n-1}$ for which $(y, n) \in I$ and $\beta_{(y,n)} \neq 0$. It therefore follows that

$$\psi(\bar{\mu}^*) = \varphi(\bar{\mu}^0) = \varphi(\delta_{(0,1)}) = \beta_{(0,1)} = 1 - \alpha.$$

Expectation of y upon termination. Here we consider the following functional:

$$\psi(\bar{\mu}^*) = \sum_{(y,n)\in T} y\bar{\mu}^*_{(y,n)}$$

which is extended to

$$\varphi(\bar{\mu}) = \sum_{(y,n)\in I} \beta_{(y,n)}\bar{\mu}_{(y,n)} + \sum_{(y,n)\in T} y\bar{\mu}_{(y,n)},$$

which has to satisfy the same invariance equations

$$\tfrac{1}{2}\beta_{(y,n+1)} + \tfrac{1}{2}\beta_{(y+1/2^n,n+1)} = \beta_{(y,n)}, \qquad (y, n) \in I.$$

Let us guess the following solution:

$$\beta_{(y,n)} = \begin{cases} 0, & y < \alpha - \dfrac{1}{2^{n-1}}, \\ A_n, & \alpha - \dfrac{1}{2^{n-1}} \leqq y < \alpha, \\ y, & \alpha \leqq y. \end{cases}$$

(Note that $\beta_{(y,n)} = A_n$ for exactly one $y \in D_{n-1}$.) Let us check the recurrence equations: If $\beta_{(y,n)} = 0$, $y < \alpha - 1/2^{n-1}$, then both y and $y + 1/2^n$ are less than $\alpha - 1/2^n$, so that the left-hand side is also 0. Suppose then that $\alpha - 1/2^{n-1} \leqq y < \alpha$. Two cases are possible.

Case I. $\alpha - 1/2^n \leqq y < \alpha$, which happens if the nth digit of α is 0 (assume an infinite binary representation of α). Then $\beta_{(y,n+1)} = A_{n+1}$ and $\beta_{(y+1/2^n,n+1)} = y + 1/2^n$. Hence we have

$$\frac{1}{2} A_{n+1} + \frac{1}{2}\left(y + \frac{1}{2^n}\right) = A_n \quad \text{if } \alpha_n = 0.$$

Case II. $\alpha - 1/2^{n-1} \leqq y < \alpha - 1/2^n$, which happens if $\alpha_n = 1$. Then $\beta_{(y,n+1)} = 0$ and $\beta_{(y+1/2^n,n+1)} = A_{n+1}$. Hence

$$\tfrac{1}{2} A_{n+1} = A_n \quad \text{if } \alpha_{n=1}.$$

Combining both cases, we can write

$$A_{n+1} = 2A_n - (1 - \alpha_n)\left(y_n + \frac{1}{2^n}\right).$$

where y_n is the binary fraction represented by the first $(n-1)$ digits of α, i.e.

$$y_n = \sum_{j=1}^{n-1} \frac{\alpha_j}{2^j}.$$

The solution of the general equation

$$A_{n+1} = 2A_n - C_n, \qquad n \geqq 1,$$

is given by

$$A_n = 2^{n-1}\left[A_1 - \sum_{k=1}^{n-1} \frac{C_k}{2^k}\right].$$

Hence, as usual, we have to choose

$$A_1 = \sum_{k=1}^{\infty} \frac{C_k}{2^k}.$$

Hence, the expectation of y is $\beta_{(0,1)} = A_1$ (note that $(\bar{v}_1 \cdot \bar{\beta}_1) < \infty$ in this case too)

$$= \sum_{k=1}^{\infty} \frac{1}{2^k}(1 - \alpha_k)\left(y_k + \frac{1}{2^k}\right) = \underbrace{\sum_{k=1}^{\infty} \frac{1 - \alpha_k}{2^{2k}}}_{Q} + \underbrace{\sum_{k=1}^{\infty} \frac{1 - \alpha_k}{2^k} \sum_{j=1}^{k-1} \frac{\alpha_j}{2^j}}_{S}.$$

To compute this sum, we use the following equality:

$$(1-\alpha)^2 = \sum_{k,j=1}^{\infty} \frac{(1-\alpha_j)(1-\alpha_k)}{2^j 2^k} = 2 \sum_{j<k} \frac{(1-\alpha_k)(1-\alpha_j)}{2^k 2^j} + \sum_{k=1}^{\infty} \frac{1-\alpha_k}{2^{2k}}$$

$$= -2S + Q + 2 \sum_{k=1}^{\infty} \frac{1-\alpha_k}{2^k} \left(\sum_{j=1}^{k-1} \frac{1}{2^j} \right)$$

$$= -2S + Q + 2(1-\alpha) - 4Q = -2S - 3Q + 2(1-\alpha).$$

Hence

$$\beta_{(0,1)} = S + Q = \frac{2(1-\alpha) - (1-\alpha)^2 - Q}{2} = \frac{1-\alpha^2}{2} - \frac{Q}{2}.$$

Thus the derived expectation of y is less than the expectation of y under uniform distribution of y in $(\alpha, 1]$ by $Q/2$, where

$$Q = \sum_{k=1}^{\infty} \frac{1-\alpha_k}{2^{2k}}.$$

Example 5. Maximum component in a random sequence. Finally we give an example of an average-case analysis of a nonprobabilistic program, using our methods. This example will require that we deal with continuous distributions; however, our methods can be easily extended to the continuous case, as will be demonstrated below, although we will not justify this extension formally. This example is considered by Knuth [KN] and is analyzed by Ramshaw [RA] by his system of "frequentistic" assertions. Consider the following program, which finds the maximum among n random elements, all drawn independently from a uniform distribution on $[0, 1]$ (λ denotes the Lebesgue measure on $[0, 1]$; note that here we allow draws out of a continuous distribution):

$M :=$ random (λ); $C := 0$;
for $J := 2$ to n **do**
  if $(t :=$ random $(\lambda)) > M$ **then** $C := C+1$; $M := t$; **fi**
od.

C is a counter variable added to the program in order to measure the number of assignments to M. This number, the number of "left-to-right" maxima occurring in the sequence, is the performance parameter that we wish to estimate. The program states can be compactly represented by the values c, m, j of C, M, J respectively at entrance to the loop. (We will use the convention that $J = n+1$ designates terminal states.) Furthermore, since m varies over a continuous distribution, we will regard $\bar{\mu}^0$ and $\bar{\mu}^*$ as density functions in m, and as distributions in c and j. (This is the first example of a continuous distribution; the results obtained so far can be easily generalized to this case, by simply replacing sums by the appropriate integrals.)

We thus wish to compute

$$\psi(\bar{\mu}^*) = \int_0^1 \sum_{c=0}^{n-1} c\mu^*_{c,n+1}(m)\, d\lambda(m).$$

We extend ψ to an invariant functional φ over S, so that

$$\varphi(\bar{\mu}) = \sum_{j=2}^{n} \sum_{c=0}^{n-1} \int_0^1 \gamma_{c,j}(m)\mu_{c,j}(m)\, d\lambda(m) + \sum_{c=0}^{n-1} c \int_0^1 \mu_{c,n+1}(m)\, d\lambda(m).$$

In a continuous model, the transition probability matrix has to be written as a kernel representing "transition density" in the continuous parameter m (see Revuz [RV] for details). Thus, we have the following nonzero entries:

$$\left.\begin{aligned} P_{(c,j,m),(c,j+1,u)} &= m\delta_m(u), \\ P_{(c,j,m),(c+1,j+1,u)} &= \chi_{(m,1]} \cdot \lambda(u), \end{aligned}\right\} \quad j \leqq n,$$
$$P_{(c,n+1,m),(c,n+1,u)} = \delta_m(u).$$

The first line describes an iteration step of the loop at which C has not been incremented, so that m does not change; the total weight of this transition is m. The second line describes an iteration step at which C is incremented, and the new value (u) of m is greater than the old value. The third line describes terminating "transitions".

The invariance of φ requires the following recurrence equations to hold:

$$\gamma_{c,j}(m) = m\gamma_{c,j+1}(m) + \int_m^1 \gamma_{c+1,j+1}(u)\, d\lambda(u), \qquad j \leqq n,$$

where $\gamma_{c,n+1}(m) = c$. We will prove that

$$\gamma_{c,n+1-r}(m) = c + \sum_{i=1}^{r} \frac{1-m^i}{i},$$

by induction on r. The equality holds for $r = 0$. Assume it holds for some r. Then

$$\begin{aligned} \gamma_{c,n-r}(m) &= m\gamma_{c,n+1-r}(m) + \int_m^1 \gamma_{c+1,n+1-r}(u)\, d\lambda(u) \\ &= m\left[c + \sum_{i=1}^{r} \frac{1-m^i}{i}\right] + \int_m^1 \left[c+1 + \sum_{i=1}^{r^*} \frac{1-u^i}{i}\right] du \\ &= mc + m\sum_{i=1}^{r} \frac{1-m^i}{i} + (1-m)c + (1-m) + \sum_{i=1}^{r} \frac{(1-m) - (1-m^{i+1})/(i+1)}{i} \\ &= c + \sum_{i=1}^{r} \frac{1-m^{i+1}}{i+1} + 1 - m = c + \sum_{i=1}^{r+1} \frac{1-m^i}{i}. \end{aligned}$$

Note that here φ is uniquely determined, and so must satisfy (V1)–(V3). Hence $\varphi(\bar{\mu}^0) = \psi(\bar{\mu}^*)$ where $\bar{\mu}^0$ is the initial distribution on entry to the loop. This initial distribution, however, is concentrated on $c = 0$ and $j = 2$ and is uniformly distributed in m. Hence we have

$$\begin{aligned} \psi(\bar{\mu}^*) = \int_0^1 \gamma_{0,2}(m)\, d\lambda(m) &= \int_0^1 \sum_{i=1}^{n-1} \frac{1-m^i}{i}\, d\lambda(m) \\ &= \sum_{i=1}^{n-1} \frac{1 - 1/(i+1)}{i} = \sum_{i=2}^{n} \frac{1}{i} = H_n - 1, \end{aligned}$$

which is the result in [KN, § 1.2.10].

REFERENCES

[CH] K. L. CHUNG, *Markov Chains with Stationary Transition Probabilities*, Springer-Verlag, New York, 1967.

[FL] R. W. FLOYD, *Assigning meaning to programs*, in Proc. Mathematical Aspects of Computer Science, J. T. Schwartz, ed., American Mathematical Society, New York, 1967, pp. 19–32.

[HS] S. HART AND M. SHARIR, *Concurrent probabilistic programs, or how to schedule if you must,* Technical Report, School of Mathematical Sciences, Tel Aviv Univ., Tel Aviv, Israel, 1982.

[HS2] ——, *Propositional probabilistic temporal logics,* Technical Report, School of Mathematical Sciences, Tel Aviv Univ., Tel Aviv, Israel, 1983.

[HSP] S. HART, M. SHARIR AND A. PNEULI, *Termination of probabilistic concurrent programs,* Proc. 9th ACM Symposium on Principles of Programming Languages, Association for Computing Machinery, New York, 1982, pp. 1–7; ACM Trans. Prog. Languages and Systems, 5 (1983).

[KSK] J. G. KEMENY, J. L. SNELL AND A. W. KNAPP, *Denumerable Markov Chains,* 2nd ed., Springer-Verlag, New York, 1976.

[KN] D. E. KNUTH, *The Art of Computer Programming,* Vol. I, Addison-Wesley, Reading, MA, 1968.

[KO] D. KOZEN, *Semantics of probabilistic programs,* in 20th IEEE Symposium on the Foundations of Computer Science, Institute of Electronics and Electrical Engineers, New York, 1979, pp. 101–114.

[LR] D. LEHMANN AND M. O. RABIN, *On the advantages of free choice,* in Proc. 8th ACM Symposium on Principles of Programming Languages, Association for Computing Machinery, New York, 1981, pp. 133–138.

[LS] D. LEHMANN AND S. SHELACH, *Reasoning with time and chance,* Technical Report, The Hebrew Univ., Jerusalem, Israel, 1982.

[MW] J. MORRIS AND B. WEGBREIT, *Subgoal induction,* Comm. ACM, 20 (1977), pp. 209–222.

[PA] D. PARK, *Fixpoint induction and proofs of program properties,* in Machine Intelligence, B. Meltzer and D. Michie, eds., Vol. 5, Edinburgh Univ. Press, Edinburgh, 1969, pp. 59–78.

[RA] L. H. RAMSHAW, *Formalizing the analysis of algorithms,* Ph.D. thesis, Report STAN-CS-79-742, Department of Computer Science, Stanford Univ., Stanford, CA, June, 1979.

[RB] M. O. RABIN, *Probabilistic algorithms,* in Algorithms and Complexity—New Directions and Recent Results, J. F. Traub, ed., Academic Press, New York, 1976, pp. 21–40.

[RE] J. H. REIF, *Logics for probabilistic programs,* in Proc. 12th ACM Symposium on the Theory of Computing, Association for Computing Machinery, New York, 1980, pp. 8–13.

[RM] C. V. RAMAMOORTHY, *Discrete Markov analysis of computer programs,* in Association for Computing Machinery, 20th National Conference, Cleveland, Association for Computing Machinery, New York, pp. 386–392.

[RV] D. REVUZ, *Markov Chains,* North-Holland, Amsterdam, 1975.

[SD] N. SAHEB-DJAHROMI, *Probabilistic* LCF, in Proc. Conference on the Mathematical Foundations of Computer Science, Springer-Verlag, New York, 1978, pp. 442–451.

[WE] B. WEGBREIT, *Verifying program performance,* J. Assoc. Comput. Mach., 23 (1976), pp. 691–699.