Probabilistic Hoare-like Logics in Comparison

Miguel D. Vásquez[1] **, Nicolás Wolovick**[1] **, Pedro R. D'Argenio**[1*]

[1]Fac. de Matemática, Astronomía y Física, Universidad Nacional de Córdoba,
Ciudad Universitaria, 5000, Córdoba, Argentina

{dargenio,nicolasw}@famaf.unc.edu.ar, mvasquez@hal.famaf.unc.edu.ar

Abstract. *Probabilistic algorithms are recognized for their simplicity and speed. A canonical example is the Miller-Rabin primality test algorithm. It is simple and achieves high accuracy with a small amount of computation. In this paper, we present two verification exercises of this algorithm using two different approaches: one being a probabilistic extension of the weakest precondition calculus [17, 15]. and the other, a probabilistic extension of Hoare logic [6, 7] We define verification strategies/patterns and establish comparisons between the logics, stressing their strengths and weaknesses.*
Keywords*: probabilistic program verification, Hoare logic, Miller-Rabin primality test, weakest pre-expectations, pGCL, probabilistic logic, pH.*

1. Introduction

Probabilistic algorithms are widely recognized for their simplicity, elegance and performance. They even provide solutions to problems unsolvable in a non-probabilistic setting [5]. Probabilistic sequential programs usually add to standard programming languages either a probabilistic choice or a probabilistic assignment [18]. In the 70's, Rabin introduced two sequential randomized algorithms showing a good accuracy-efficiency trade-off [20]. One of these examples was the Miller-Rabin primality test. This test allows to decide primality, getting a false positive with a probability that decreases exponentially to zero with the number of repetitions of a basic test. The importance of this algorithm is widely recognized in cryptography and number theory applications, because primes of hundreds of digits could be effectively found. (Although this problem has recently been shown to be polynomial [1], Miller-Rabin is by far the *de-facto* standard for primality test.) Formal verification of this kind of algorithms (so-called *one-sided Monte Carlo*) is important since an incorrect answer does not necessarily imply an incorrect algorithm and any kind of testing must be statistical [6, 10].

Assertional reasoning allows the verification of programs directly in a state-less manner, abstracting away from actual execution steps. As a consequence generic and parametric algorithms like the one under study, can be verified at no extra cost, outperforming any kind of automatic state-exploration model-checking technique. Probabilistic programs are probabilistic state transformers, changing the "distribution cloud" as execution goes on [13], therefore traditional assertional verification is no longer valid because in general a predicate in a probabilistic state cannot be evaluated to a truth value. The logic have to be lifted in some way to cope with state distributions in order to verify, for example, that a given algorithm end up satisfying "$0 < x$ with a probability greater than 1/2".

There are two Hoare-like logics suitable to verify postconditions like the former. In [17, 15, 16], Morgan et al. defined a weakest pre-expectation calculus (named $pGCL$)

[*]CONICET Researcher. Supported by the ANPCyT project PICT 11-11738 and the EC Project IST-2001-35304, AMETIST

generalizing Dijkstra's weakest precondition calculus [3]. Den Hartog [6, 7] proposed a probabilistic logic and a sound and complete pre/postcondition calculus (called pH) that extends Hoare logic [8]. Both assertional-based methods deal with some minor variations of Dijkstra's Guarded Command Language GCL that includes the probabilistic choice construction $S_0 \oplus_p S_1$ that selects S_0 with probability p and S_1 with probability $1 - p$.

In this paper we use both logics on the verification of the Miller-Rabin primality test with the intention of learning their strengths and weaknesses while comparing them. This process has two derived aims. On the one hand, the exercises of verification help to develop new theorems and techniques that ease proving and facilitates reading. On the other hand, we expect to learn from this two theories in order to further develop verification techniques that applies to more general types of algorithms (such as those including concurrency).

Previous work. Hurd [10] reported a full verification of the same algorithm presented here using his framework early defined in [9]. He presented a functional version of a concrete imperative algorithm appearing in [2]. The verification is based on the functional representation and completely achieved using a theorem prover, including all number theory aspects of the problem. To our knowledge, no other formal verification of this algorithm exists, however there are many assertional-based verifications of sequential and concurrent probabilistic algorithms. The majority of them are based on $pGCL$ and carried out by the same developers of the method (e.g. [16, 4, 11]). To our knowledge, this is the first time that den Hartog's logic is used outside the verification of toy algorithms.

Organization of the paper. Section 2 introduces the Miller-Rabin algorithm. Section 3 presents $pGCL$ and its calculus and reports the verification of the algorithm. Section 4 follows the same structure but for pH. Section 5 concludes the paper with a discussion and comparison of both verification techniques.

2. The Algorithm

The input is taken from variable n, containing the number to test for primality, and t, that bounds the number of times the basic test is performed. The algorithm includes initialization phase $FactorTwos$ that decomposes $n-1$ into r, s such that $n-1 = 2^r s$ for an odd s. They are needed through all iterations of the main loop that follows. The basic test $Uniform; Witness$ is repeated t times, setting $witness$ if number a, randomly chosen from a uniform distribution in the integer interval $[1, n–1]$, is witness of the compositeness of n. We later formalize this fact, meanwhile it is only important to understand that if a is a witness of n, then n is composite. Non-compositeness is accumulated in $prime$. If the loop finishes, then no witness of compositeness was found and two things may have happened: either n is prime or, even when n is composite, no appropriate witnesses was (randomly) chosen. The *abundance of witnesses* for non-prime numbers ensures that the probability of getting a false positive decreases exponentially on t by $1/2^t$. We later formalize this specification and prove that the algorithm satisfies it.

3. $pGCL$ Calculus

3.1. Background

In [17] Morgan et al. extended the wp-calculus integrating probabilistic and non-deterministic choice. Instead of predicates, $pGCL$ wp-calculus transforms *expectations*, real expressions from state variables to $[0, 1]$. Usual boolean operators are lifted in the expectation calculus in order to operate the interval of $[0, 1]$. Conjunction becomes minimum, disjunction the maximum and negation is the probability of the complementary event. In

```
$r := 0;$
$s := n-1;$
do $s \ mod \ 2 = 0 \rightarrow$                              } $FactorTwos$
    $s := s \ div \ 2;$
    $r := r + 1$
od;
$prime := true;$
do $t \neq 0 \rightarrow$
    $a := n;$
    do $\neg(0 < a < n) \rightarrow$
        $k := n;$
        $a := 0;$
        do $k \neq 0 \rightarrow$                               } $Uniform'$    } $Uniform$
            $a := 2a \oplus_{1/2} a := 2a + 1;$
            $k := k \ div \ 2$
        od
    od;
    $a' := a^s \ mod \ n;$
    $y := 0;$
    $witness := false;$
    do $y \neq r \wedge \neg witness \rightarrow$
        if $a' \neq 1 \wedge a' \neq n-1$ then
            $a' := a'^2 \ mod \ n;$
            $witness := (a' = 1)$                          } $Witness$
        else $a' := a'^2 \ mod \ n$
        fi ;
        $y := y + 1;$
    od;
    $witness := witness \vee (a' \neq 1);$
    $prime := prime \wedge \neg witness;$
    $t := t - 1$
od
```

Figure 1: $MillerRabin.n.t$

order to ease its interpretation the symbols $\sqcap, \sqcup, \bar{\cdot}$ are used instead. Regular predicates, also called *deterministic*, lift to expectations with the operator $[\cdot]$, giving 1 if the predicate is $true$, otherwise 0. Expectations giving values in $\{0, 1\}$ are called *standard*. The following table summarizes wp-calculus of pre-expectations. In the probabilistic choice p can be an expression involving variables, so conditional choice is a special case of the earlier.

$$\begin{array}{rcl}
wp.(\mathbf{skip}).R & \equiv & R \\
wp.(x := E).R & \equiv & R[x := E] \\
wp.(S;T).R & \equiv & wp.S.(wp.T.R) \\
wp.(S \oplus_p T).R & \equiv & p * wp.S.R + (1-p) * wp.T.R \\
wp.(\mathbf{if}\ B\ \mathbf{then}\ S\ \mathbf{else}\ T\ \mathbf{fi}).R & \equiv & [B] * wp.S.R + [\neg B] * wp.T.R
\end{array}$$

$$\frac{[G] * I \Rrightarrow wp.S.I \quad I \Rrightarrow wp.(\mathbf{do}\ G\ \rightarrow S\ \mathbf{od}).1}{I \Rrightarrow wp.(\mathbf{do}\ G\ \rightarrow S\ \mathbf{od}).([\neg G] * I)}$$

The first constraint in the antecedent of the rule for loop construction states the invariance of predicate I. The second one, states that the invariant must imply termination of the loop where predicate $wp.(\mathbf{do}\ G\ \rightarrow S\ \mathbf{od}).1$ is the expectation that the loop finishes. We will in general use T to denote this last predicate.

For example, we calculate $wp.(x := 0 \oplus_{1/2} x := 1).([x = 0]) \equiv 1/2$ which means that the minimum probability of ending in a state validating $x = 0$ is greater or equal than $1/2$. Calculations may also reduce to a non-standard expectation as in: $wp.(x := 0 \oplus_{1/3} x := 1).([x = y]) \equiv 1/3 * [0 = y] + 2/3 * [1 = y]$. This predicate should be read as "the minimum probability of reaching a state satisfying $x = y$ is a function of y, giving $1/3$ if $y = 0$, $2/3$ if $y = 1$ and 0 otherwise".

The equivalence symbol ($\equiv$) used above is lifted to functions as point to point equality. Implication and consequence for expectations (respectively denoted by $\Rrightarrow$, and $\Lleftarrow$) are defined pointwise from $\leq$ and $\geq$ respectively. Below we summarize some important properties that hold in the wp-calculus [16].

$$\begin{array}{lr} wp.S.(a * R + b * R' \ominus c) \Lleftarrow a * (wp.S.R) + b * (wp.S.R') \ominus c & \textbf{\textit{sublinearity}} \\ wp.S.R \Rrightarrow wp.S.R' \qquad \text{if } R \Rrightarrow R' & \textbf{\textit{monotonicity}} \\ wp.S.(a * R) \equiv a * (wp.S.R) & \textbf{\textit{scaling}} \\ wp.S.(R \mathbin{\&} R') \Lleftarrow wp.S.R \mathbin{\&} wp.S.R' & \textbf{\textit{subconjunctivity}} \end{array}$$

where $a \ominus b \doteq (a - b)\ max\ 0$ is the bounded subtraction and $R \mathbin{\&} R' \doteq a + b \ominus 1$ is one possible lifting for boolean conjunction[1].

The following theorems will be handy to check loop correctness.

Theorem 1 (Probabilistic Variant Rule) *[15, 16] Let V be an integer expression over program variables, such that: 1) it is bounded from above and from below, 2) there is a fixed probability $p \neq 0$ such that V is decreased in each iteration. Then the probability of loop termination is 1.*

Lemma 1 (0-1 Termination Law) *[15] Let I be an invariant of the loop, if $0 < \rho$ and $\rho * I \Rrightarrow T$ then $I \Rrightarrow T$.*

Theorem 2 (Deterministic Loop Theorem) *[21]*

$$wp.(\mathbf{do}\ G \ \rightarrow\ S\ \mathbf{od}).Q \equiv \textstyle\sum_{j=0}^{\infty} h^j.([\neg G] * Q) \quad \textit{where} \quad h.\gamma \doteq [G] * wp.S.\gamma$$

3.2. Correctness Proof

We split the verification focusing in the three blocks of figure 1: $FactorTwos$, $Uniform$ and $Witness$. Next we join all the pieces considering the cases of n being prime and not prime. We use predicate $\mathsf{PRIME}.n$ to indicate that n is prime.

$FactorTwos$**.** The first part is devoted to calculate r and s, such that $n-1 = 2^r s$ and s is odd. This section of the code is non-probabilistic, also called *standard*, and under this condition $pGCL$ agrees with GCL. As a consequence, we omit calculations of this part of the program and state the following without proof:

$$1 \Rrightarrow wp.FactorTwos.[n-1 = 2^r s \wedge \mathsf{odd}.s]$$

$Uniform$**.** This code section generates a uniformly distributed random number in the interval $[1, n-1]$ by means of a fair coin. Following [9] we pick a random number in $[0, 2^{\mathsf{lg2}.n})$ until one is chosen in the desired interval $[1, n-1]$. Note that this procedure may loop an unbounded number of times, however the *expected* time is polynomial. (In [10] a true polynomial time technique, hence in $co-RP$, is presented.) The function $\mathsf{lg2}.n \doteq \lceil log_2(n+1) \rceil$ counts the number of bits needed to encode n.

[1]For example $a \sqcap b$ and $\lfloor a * b \rfloor$ are also valid.

The inner loop $\mathit{Uniform}'$ is the responsible for throwing a fair coin and accumulating in a the decimal representation of the toss sequence. This is the invariant that captures the core of this loop.

$$I : [a2^{\mathsf{lg}2.k} \leq i < (a+1)2^{\mathsf{lg}2.k}] * 2^{-\mathsf{lg}2.k}$$

The predicate between square brackets can also be written as $\frac{i}{2^{\mathsf{lg}2.k}} - 1 < a \leq \frac{i}{2^{\mathsf{lg}2.k}}$ and agrees with $a = i \ div \ (2^{\mathsf{lg}2.k})$ under integer division. Therefore, the intuition behind I is that "the probability that a represents the $\mathsf{lg}2.k$ most significant bits of i is $2^{-\mathsf{lg}2.k}$". In the initialization we easily obtain the required pre-expectation using substitution[2].

$$wp.(k, a := n, 0).I \equiv [0 \leq i < 2^{\mathsf{lg}2.n}] * 2^{-\mathsf{lg}2.n}$$

As k is a valid probabilistic variant, establishing termination with probability 1, we focus on showing that the loop body maintains I.

$$\begin{array}{ll}
 & wp.((a := 2a \oplus_{1/2} a := 2a+1); k := k \ div \ 2).I \\
\equiv & \{ \text{ sequential composition, assignment, log properties } \} \\
 & wp.(a := 2a \oplus_{1/2} a := 2a+1).\left(\left[a2^{(\mathsf{lg}2.k)-1} \leq i < (a+1)2^{(\mathsf{lg}2.k)-1}\right] * 2^{-(\mathsf{lg}2.k)-1}\right) \\
\equiv & \{ \text{ probabilistic choice, assignment, arithmetic } \} \\
 & 1/2 * \left(\left[a2^{\mathsf{lg}2.k} \leq i < (2a+1)2^{(\mathsf{lg}2.k)-1}\right] * 2^{-(\mathsf{lg}2.k)-1}\right) + 1/2 * \left(\left[(2a+1)2^{(\mathsf{lg}2.k)-1} \leq i < (a+1)2^{\mathsf{lg}2.k}\right] * 2^{-(\mathsf{lg}2.k)-1}\right) \\
\equiv & \{ \text{ arithmetic } \} \\
 & \left(\left[a2^{\mathsf{lg}2.k} \leq i < (2a+1)2^{(\mathsf{lg}2.k)-1}\right] + \left[(2a+1)2^{(\mathsf{lg}2.k)-1} \leq i < (a+1)2^{\mathsf{lg}2.k}\right]\right) * 2^{-\mathsf{lg}2.k} \\
\equiv & \{ \text{ expectation calculus } \} \\
 & \left[a2^{\mathsf{lg}2.k} \leq i < (a+1)2^{\mathsf{lg}2.k}\right] * 2^{-\mathsf{lg}2.k}
\end{array}$$

Finally, by the rules for probabilistic loops and sequential composition:

$$[0 \leq i < 2^{\mathsf{lg}2.n}] * 2^{-\mathsf{lg}2.n} \equiv wp.\mathit{Uniform}'.([k = 0] * I)$$

It is straightforward to prove that $[\neg(k \neq 0)] * I \Rrightarrow [a = i]$. By monotonicity, we obtain:

$$[0 \leq i < 2^{\mathsf{lg}2.n}] * 2^{-\mathsf{lg}2.n} \Rrightarrow wp.\mathit{Uniform}'.[a = i]$$

i.e., if i is in $[0, 2^{\mathsf{lg}2.n})$, the expectation of $\mathit{Uniform}'$ to set a to i is $2^{-\mathsf{lg}2.n}$, and 0 otherwise.

The outer loop picks random numbers in $[0, 2^{\mathsf{lg}2.n})$ until there is one in $[1, n-1]$, discarding those out of range. We calculate the weakest pre-expectation of the loop body under termination condition $\neg G$:

$$\begin{array}{ll}
 & wp.\mathit{Uniform}'.[0 < a < n] \\
\Lleftarrow & \{ \text{ range to disjoint union, expectation calculus, sublinearity } \} \\
 & \sum_{i=1}^{n-1} wp.\mathit{Uniform}'.[a = i] \\
\Lleftarrow & \{ \text{ previous inner body result } \} \\
 & (n-1)2^{-\mathsf{lg}2.n} * 1
\end{array}$$

Setting $\rho : (n-1)2^{-\mathsf{lg}2.n}$ and $I : 1$, we calculate[3].

$$\begin{array}{llcll}
 & \rho * I & & & \\
\Rrightarrow & \{ \text{ previous result } \} & & & wp.\mathit{Uniform}'.T \\
 & wp.\mathit{Uniform}'.[\neg G] & & \Rrightarrow & \{ \text{ inner loop is terminating } \} \\
\Rrightarrow & \{ [\neg G] \Rrightarrow T, \text{ monotonicity } \} & & & T
\end{array}$$

By 0-1 Termination Law, $I \Rrightarrow T$. Therefore $T \equiv 1$ and $\mathit{Uniform}$ is always terminating.

[2] In general, we use multiassignments. Decoupled assignments can be proved in a similar way [22].

[3] This proof rests on a pair of small proofs, namely $[\neg G] \Rrightarrow T$ and $wp.body.T \Rrightarrow T$ easily provable from the semantic fact $wp.loop.Q \equiv [G] * wp.body.(wp.loop.Q) + [\neg G]$.

For partial correctness, we use the Deterministic Loop Theorem to calculate the greatest pre-expectation of $Uniform$ for a particular value of a. Therefore, we calculate each term of the summation given in the theorem. The first term is $[0 < a < n] * [a = i]$. For the second one, we calculate

$$\begin{array}{ll}
 & h.([0 < a < n] * [a = i]) \\
\equiv & \{ \text{definition} \} \\
 & [\neg(0 < a < n)] * wp.Uniform'.([0 < a < n] * [a = i]) \\
\equiv & \{ \text{expectation and predicate calculus} \} \\
 & [\neg(0 < a < n)] * wp.Uniform'.([0 < i < n] \mathbin{\&} [a = i]) \\
\Leftarrow & \{ \text{subconjunctivity} \} \\
 & [\neg(0 < a < n)] * (wp.Uniform'.[0 < i < n] \mathbin{\&} wp.Uniform'.[a = i]) \\
\Leftarrow & \{ \text{orthogonality}^4, \text{previous result} \} \\
 & [\neg(0 < a < n)] * \left([0 < i < n] \mathbin{\&} 2^{-\mathsf{lg2}.n}\right) \\
\equiv & \{ \text{expectation calculus} \} \\
 & [\neg(0 < a < n)] * [0 < i < n] * 2^{-\mathsf{lg2}.n}
\end{array}$$

Using the result $(1 - \rho) \Rrightarrow wp.Uniform'.[\neg(0 < a < n)]$, and previous calculation as a base case we prove by induction (for full proofs refer to [22]) :

$$h^j.([0 < a < n] * [a = i]) \equiv [\neg(0 < a < n)] * [0 < i < n] * \rho/(n-1) * (1 - \rho)^{j-1}, \qquad 1 \leq j$$

The weakest pre-expectation can be readily calculated using deterministic loop theorem.

$$\begin{array}{ll}
 & wp.(\textbf{do}\ \neg(0 < a < n) \rightarrow Uniform'\ \textbf{od}).[a = i] \\
\equiv & \{ \text{deterministic loop theorem} \} \\
 & \sum_{j=0}^{\infty} h^j.([0 < a < n] * [a = i]) \\
\equiv & \{ \text{previous results, calculus} \} \\
 & [0 < a < n] * [a = i] + [\neg(0 < a < n)] * [0 < i < n] * \rho/(n-1) * \sum_{j=0}^{\infty}(1 - \rho)^j \\
\equiv & \{ \text{mathematical fact } \rho \in (0,1) \Rightarrow \sum_{j=0}^{\infty}(1 - \rho)^j = 1/\rho \} \\
 & [0 < a < n] * [a = i] + [\neg(0 < a < n)] * [0 < i < n] * 1/(n-1)
\end{array}$$

Applying sequential composition and assignment in the above result, in order to add loop initialization $a := n$, we get the required pre-expectation for $Uniform$:

$$[0 < i < n] * 1/(n-1) \Rrightarrow wp.Uniform.[a = i] \tag{1}$$

Invariant given by Morgan for $Uniform$:
Let be $\$n := 2^{\mathsf{lg2}.n}$, and the probabilistic invariant:

$$I := [\neg(0 < a < n)] * [0 \leq i < n]/(n-1) \; + \; [0 < a < n] * [a = i]$$

so the initalization will be: $wp.(a := n).I \equiv [0 < i < n]/(n-1)$
and the exit of the loop will be: $[0 < a < n] * I \Rrightarrow [a = i]$
Remember that $[0 \leq i < \$n]/\$n \Rrightarrow wp.Uniform'.[a = i]$

[4] $wp.S.R \equiv R$ if $var(S)$ and $var(R)$ are disjoint.

Now we show that I es loop-invariant.

$$
\begin{array}{ll}
& wp.\mathit{Uniform}'.I \\
\Lleftarrow & \{ \text{ sublinearity; predicate calculus } \} \\
& [0<i<n]/(n{-}1) * wp.\mathit{Uniform}'.[\neg(0<a<n)] \ + \ wp.\mathit{Uniform}'.([0<a<n]\&[a=i]) \\
\Lleftarrow & \{ \text{ subconjunctivity } \} \\
& [0<i<n]/(n{-}1) * wp.\mathit{Uniform}'.[\neg(0<a<n)] \ + \ wp.\mathit{Uniform}'.[0<a<n] \ \& \ wp.\mathit{Uniform}'.[a=i] \\
\Lleftarrow & \{ \text{ calculate } wp \ \} \\
& [0<i<n]/(n{-}1) * (\$n-n+1) * [0\le i<\$n]/\$n \quad + \quad (n{-}1) * [0<i<\$n]/\$n \ \& \ [0<i<\$n]/\$n \\
\Lleftarrow & \{ \text{ expectation calculus } \} \\
& [0<i<n]/(n{-}1) \\
\Lleftarrow & \{ \text{ expectation calculus } \} \\
& [\neg(0<a<n)] * I
\end{array}
$$

Finally we have: $\quad [0<i<n]/(n{-}1) \ \Rrightarrow \ wp.\mathit{Uniform}.[a=i]$

$\mathit{Witness}$**.** The second part is the core of the Miller-Rabin algorithm. The verification condition is the following:

$$1 \Rrightarrow wp.\mathit{Witness}.[\mathit{witness} \Leftrightarrow \mathsf{WITNESS}.n.a] \tag{2}$$

where predicate $\mathsf{WITNESS}$ is defined by:

$$
\begin{array}{rcl}
\mathsf{WITNESS}.n.a & \doteq & a^{n-1}\, mod\, n \neq 1 \lor \mathsf{WITNESS}'.n.a.y \\
\mathsf{WITNESS}'.n.a.y & \doteq & \left(\exists j : 0\le j<y : a^{2^{j+1}s}\, mod\, n{=}1 \land a^{2^j s}\, mod\, n{\neq}1 \land a^{2^j s}\, mod\, n{\neq}n{-}1\right)
\end{array}
$$

where s is such that $n{-}1 = 2^r s$, for some r, and $\mathsf{odd}.s$. $\mathsf{WITNESS}$ left disjunct is *Fermat's little theorem.* The right disjunct is to cope with *Carmichael numbers* and ensure that at least $(n-1)/2$ values of a are *witnesses* of n being composite (see [2, 5, 18, 10] and property (4)).

The invariant specifies the incremental computation of this existential quantification and the value of auxiliary variable a'.

$$I \ : \ \left[a' = a^{2^y s}\ mod\ n \land \mathit{witness} = \mathsf{WITNESS}'.n.a.y\right]$$

The correctness proof for this loop is more involved than the others, but as it is non-probabilistic, we use standard techniques from [3]. The proof contains, however, "expectation calculus" that make use of simple lemmas relating minimum, maximum, plus and the lift operator $[\cdot]$. Details of this proof are reported in [22].

One iteration. Since $\mathit{MillerRabin}$ is t repetitions of $\mathit{Uniform}; \mathit{Witness}$, we first deal with only one iteration. The following proofs rest on the next two number theory theorems:

$$(\forall n, a : 0<a<n : \mathsf{PRIME}.n \Rightarrow \neg\mathsf{WITNESS}.n.a) \tag{3}$$
$$2<n \land \mathsf{odd}.n \land \neg\mathsf{PRIME}.n \Rightarrow (n{-}1)/2 \le |\ \{A : 0<A<n : \mathsf{WITNESS}.n.A\}\ | \tag{4}$$

We divide this proof in two cases, depending on whether n is prime or not. If $\mathsf{PRIME}.n$ then by (3) we would like to establish $\neg\mathit{witness}$ with probability one. If $\neg\mathsf{PRIME}.n$, by (4), we show that the minimum probability of establishing $\mathit{witness}$ is strictly greater than one half, and for $\neg\mathit{witness}$ we have an expectation less than a half. The last two results will be very important for the verification of the main loop. We calculate the greatest pre-expectation for $[witness]$ if $\neg\mathsf{PRIME}.n$ as follows.

$$
\begin{array}{ll}
& wp.(\mathit{Uniform};\ \mathit{Witness}).[witness] \\
\equiv & \{ \text{ predicate calculus, expectation calculus, sequential composition } \} \\
& wp.\mathit{Uniform}.wp.\mathit{Witness}.([witness \Leftrightarrow \mathsf{WITNESS}.n.a]\ \&\ [\mathsf{WITNESS}.n.a]) \\
\Lleftarrow & \{ \text{ subconjunctivity, monotonicity } \} \\
& wp.\mathit{Uniform}.(wp.\mathit{Witness}.[witness \Leftrightarrow \mathsf{WITNESS}.n.a]\ \&\ wp.\mathit{Witness}.[\mathsf{WITNESS}.n.a]) \\
\equiv & \{ \text{ orthogonality, equation (2), expectation calculus } \} \\
& wp.\mathit{Uniform}.[\mathsf{WITNESS}.n.a] \\
\equiv & \{ \text{ let } \{A_i\}_{i=1}^{K} \text{ be all witnesses of } n \text{ in } 0<i<n\text{, by (4), } (n-1)/2 \le K \ \} \\
& wp.\mathit{Uniform}.[\bigvee_{i=1}^{K} a = A_i] \\
\equiv & \{ \text{ expectation calculus } \} \\
& wp.\mathit{Uniform}.\sum_{i=1}^{K}[a = A_i] \\
\Lleftarrow & \{ \text{ sublinearity } \} \\
& \sum_{i=1}^{K} wp.\mathit{Uniform}.[a = A_i] \\
\Lleftarrow & \{ \text{ equation (1) } \} \\
& \sum_{i=1}^{K}[0 < A_i < n] * 1/(n-1) \\
\equiv & \{ \text{ calculus, } \epsilon \doteq K/(n-1) - 1/2\text{, where } 0<\epsilon<1/2 \ \} \\
& 1/2 + \epsilon
\end{array}
$$

The other bound can be proved in a similar way, so we have

$$1/2 + \epsilon \quad \Rrightarrow \quad wp.(\mathit{Uniform};\ \mathit{Witness}).[witness] \tag{5}$$

$$1/2 - \epsilon \quad \Rrightarrow \quad wp.(\mathit{Uniform};\ \mathit{Witness}).[\neg witness] \tag{6}$$

For $\mathsf{PRIME}.n$, we use the same proof pattern, getting

$$wp.(\mathit{Uniform};\ \mathit{Witness}).[\neg witness] \equiv 1 \tag{7}$$

Outer loop. Again we split the proof in two according to n's primality. For composite n, we look for a loop invariant I whose pre-expectation respects $1-2^{-t} \Rrightarrow wp.(prime := true).I$, right before entering the loop, and that agrees with the postcondition in the loop exit $[\neg(t \neq 0)]*I \Rrightarrow [\neg prime]$. The invariant $I : [\neg prime]+[prime]*(1-2^{-t})$, captures the idea of finishing in the postcondition, or (note terms disjointness) the probability of ending up with a wrong answer. It agrees with loop initialization and exit conditions and is maintained in the loop body.

$$
\begin{array}{ll}
 & wp.(\mathit{Uniform};\ \mathit{Witness}; \mathit{prime} := \mathit{prime} \land \neg \mathit{witness}; t := t-1).I \\
\equiv & \{\ \text{sequential composition, assignment}\ \} \\
 & wp.(\mathit{Uniform};\ \mathit{Witness}).([\neg \mathit{prime} \lor \mathit{witness}] + [\mathit{prime} \land \neg \mathit{witness}] * (1 - 2^{-t+1})) \\
\equiv & \{\ \text{lifted logical connectives}\ \} \\
 & wp.(\mathit{Uniform};\ \mathit{Witness}).(([\neg \mathit{prime}] \sqcup [\mathit{witness}]) + ([\mathit{prime}] \sqcap [\neg \mathit{witness}]) * (1 - 2^{-t+1})) \\
\Lleftarrow & \{\ \text{sublinearity, scaling}\ \} \\
 & wp.(\mathit{Uniform};\ \mathit{Witness}).([\neg \mathit{prime}] \sqcup [\mathit{witness}]) \\
 & +(1 - 2^{-t+1}) * wp.(\mathit{Uniform};\ \mathit{Witness}).([\mathit{prime}] \sqcap [\neg \mathit{witness}]) \\
\Lleftarrow & \{\ \text{submaximality, subminimality}^5 \} \\
 & wp.(\mathit{Uniform};\ \mathit{Witness}).[\neg \mathit{prime}] \sqcup wp.(\mathit{Uniform};\ \mathit{Witness}).[\mathit{witness}] \\
 & +(1 - 2^{-t+1}) * (wp.(\mathit{Uniform};\ \mathit{Witness}).[\mathit{prime}] \sqcap wp.(\mathit{Uniform};\ \mathit{Witness}).[\neg \mathit{witness}]) \\
\equiv & \{\ \text{orthogonality}\ \} \\
 & [\neg \mathit{prime}] \sqcup wp.(\mathit{Uniform};\ \mathit{Witness}).[\mathit{witness}] \\
 & +(1 - 2^{-t+1}) * ([\mathit{prime}] \sqcap wp.(\mathit{Uniform};\ \mathit{Witness}).[\neg \mathit{witness}]) \\
\Lleftarrow & \{\ \text{facts (5,6)}\ \} \\
 & ([\neg \mathit{prime}] \sqcup (1/2 + \epsilon)) + (1 - 2^{-t+1}) * ([\mathit{prime}] \sqcap (1/2 - \epsilon)) \\
\Lleftarrow & \{\ \text{case analysis in } \mathit{prime}\ \} \\
 & \begin{array}{ll|ll}
 & & \equiv & \{\ [\neg \mathit{prime}] = 0, [\mathit{prime}] = 1\ \} \\
 \equiv & \{\ [\neg \mathit{prime}] = 1, [\mathit{prime}] = 0\ \} & & (1/2 + \epsilon) + (1 - 2^{-t+1})(1/2 - \epsilon) \\
 & 1 \sqcup (1/2 + \epsilon) & \equiv & \{\ \text{algebra}\ \} \\
 \equiv & \{\ 1/2 + \epsilon < 1, \sqcup \text{ properties}\ \} & & 1 - 2^{-t} + \epsilon 2^{-t+1} \\
 & 1 & \Lleftarrow & \{\ 0 < \epsilon\ \} \\
 \equiv & \{\ [\neg \mathit{prime}] = 1, [\mathit{prime}] = 0\ \} & & 1 - 2^{-t} \\
 & [\neg \mathit{prime}] + (1 - 2^{-t}) * [\mathit{prime}] & \equiv & \{\ [\neg \mathit{prime}] = 0, [\mathit{prime}] = 1\ \} \\
 & & & [\neg \mathit{prime}] + (1 - 2^{-t}) * [\mathit{prime}]
 \end{array} \\
 & [\neg \mathit{prime}] + (1 - 2^{-t}) * [\mathit{prime}] \\
\Lleftarrow & \{\ [\cdot] \le 1\ \} \\
 & [t \neq 0] * I
\end{array}
$$

Once again termination condition is certain, thanks to Probabilistic Variant Rule with variant t and lower bound 0. (Upper bound is not needed because t cannot be increased.)

For PRIME.n we propose invariant $I : [\mathit{prime}]$. Initialization gives $1 \Rrightarrow wp.(\mathit{prime} := \mathit{true}).[\mathit{prime}]$ and loop exit meets postcondition $[\neg(t \neq 0)] * [\mathit{prime}] \Rrightarrow [\mathit{prime}]$. From (7) it can be deduced that loop body maintains I. Since Probabilistic Variant Rule guarantees termination, we therefore apply the loop rule and get

$$1 \Rrightarrow wp.\mathit{MillerRabin}.[\mathit{prime}]$$

4. *pH* Calculus

4.1. Background

Predicates in $pGCL$ are usual logic predicates reinterpreted as functions from states to the interval $[0, 1]$. An alternative approach to probabilistic predicates is to extend predicate logic to allow explicit manipulation of probabilities in such a manner that a predicate is still interpreted as a function from states into the boolean lattice. Thus, for instance, one could write $\mathbb{P}(\mathit{prime}) \le 2^{-T+i}$ meaning that "the probability that program variable

[5]Subminimality stands for $wp.S.([p] \sqcap [q]) \Lleftarrow wp.S.[p] \sqcap wp.S.[q]$, submaximality is similar.

$prime$ is true is greater than or equal to 2^{-T+i} for some constant T and variable i". A more complex example is predicate $(\exists i : \mathbb{P}(prime) \leq 2^{-T+i} \wedge \mathbb{P}(t = i \wedge \neg \mathsf{PRIME}.n) = 1)$ where $prime$, t, and n are program variables, T is a constant and i is a predicate variables (later we will see this is the invariant of the main loop in *MillerRabin*). Notice the two levels of logic formulas: one within predicate $\mathbb{P}$ and the other outside it, where $\mathbb{P}$ expressions are part of the basic predicates. An important restriction is imposed to these probabilistic predicates: program variables can only occur within $\mathbb{P}$ expressions, that is why the need of variable i in the formula above (the seemingly equivalent formula $\mathbb{P}(prime) \leq 2^{-T+t} \wedge \mathbb{P}(\neg \mathsf{PRIME}.n) = 1$ is not a well formed formula).

Probability predicates are interpreted on *probabilistic states*. A probabilistic state θ is a (sub)probability distribution on so called *deterministic states*, which are usual states, i.e. functions that assign values to variables. (By subprobability we mean that the total probability mass of θ may not sum up to 1. In such a case the predicate $\mathbb{P}(true) < 1$ would be valid.) Probabilistic predicates may also have the following form: $p_0 + p_1$ that is valid in probabilistic state θ if it can be split in two parts satisfying p_0 and p_1 respectively; $\rho \cdot p$ that is valid in θ if there exists θ' where p is valid and is a ρ-scaling of θ (i.e. $\theta = \rho \cdot \theta'$); $p_0 \oplus_\rho p_1$ that is a shorthand for $\rho \cdot p_0 + (1 - \rho) \cdot p_1$; and the cut $c?p$ that is valid in θ, if there exists θ' satisfying p and equal to θ if "cut" to deterministic states satisfying c (i.e. $\theta(\sigma) =$ if (c is valid in σ) then $\theta'(\sigma)$ else 0, for all deterministic state σ).

Den Hartog [6, 7] introduced these probabilistic predicates and proposed a Hoare-like logic where p and q in triple $\{p\}s\{q\}$ are predicates of this kind. A triple $\{p\}s\{q\}$ is valid if and only if it can be proven using the proof system *pH* given in Fig. 2. Note

$$\{p\}\ \textbf{skip}\ \{p\} \quad \text{(Skip)} \qquad \{p[x/e]\}\ x := e\ \{p\} \quad \text{(Assign)}$$

$$\frac{\{p\}\ s\ \{p'\} \quad \{p'\}\ s'\ \{q\}}{\{p\}\ s; s'\ \{q\}} \quad \text{(Seq)} \qquad \frac{p' \Rightarrow p \quad \{p\}\ s\ \{q\} \quad q \Rightarrow q'}{\{p'\}\ s\ \{q'\}} \quad \text{(Cons)}$$

$$\frac{\{p\}\ s\ \{q\} \quad \{p'\}\ s\ \{q\}}{\{p \vee p'\}\ s\ \{q\}} \quad \text{(Or)} \qquad \frac{\{p\}\ s\ \{q\} \quad j \notin FV(q)}{\{\exists j : p\}\ s\ \{q\}} \quad \text{(Exists)}$$

$$\frac{\{p\}\ s\ \{q\} \quad \{p\}\ s\ \{q'\}}{\{p\}\ s\ \{q \wedge q'\}} \quad \text{(And)} \qquad \frac{\{p\}\ s\ \{q\} \quad j \notin FV(p)}{\{p\}\ s\ \{\forall j : q\}} \quad \text{(Forall)}$$

$$\frac{\{p\}\ s\ \{q\}}{\{\rho \cdot p\}\ s\ \{\rho \cdot q\}} \quad \text{(Lin}\cdot\text{)} \qquad \frac{\{p\}\ s\ \{q\} \quad \{p'\}\ s\ \{q'\}}{\{p + p'\}\ s\ \{q + q'\}} \quad \text{(Lin+)}$$

$$\frac{\{c?p\}\ s\ \{q\} \quad \{\neg c?p\}\ s'\ \{q'\}}{\{p\}\ \textbf{if}\ c\ \textbf{then}\ s\ \textbf{else}\ s'\ \textbf{fi}\ \{q + q'\}} \quad \text{(If)} \qquad \frac{\{p\}\ s\ \{q\} \quad \{p\}\ s'\ \{q'\}}{\{p\}\ s \oplus_\rho s'\ \{q \oplus_\rho q'\}} \quad \text{(Prob)}$$

$$\frac{\{p\}\ \textbf{if}\ c\ \textbf{then}\ s\ \textbf{else skip fi}\ \{p\} \qquad p \text{ is } \langle c, s\rangle\text{-closed}}{\{p\}\ \textbf{do}\ c \rightarrow s\ \textbf{od}\{p \wedge \mathbb{P}(c) = 0\}} \quad \text{(While)}$$

Figure 2: Proof system pH ($FV(p)$ denotes the set of free variables in p)

that the rules (Skip), (Assign), (Seq) y (Cons) are the same as Hoare logic, but (If) and (While) have been changed. The new rules (Or), (And), (Exists) and (Forall) are needed for completeness reasons (otherwise, valid triples with respect to denotational semantics like $\{p \vee q\}$ **skip** $\oplus_{1/2}$ **skip** $\{p \vee q\}$ could not be proven), while (Prob), (Lin $+$) and (Lin $\cdot$) captures the probabilistic choice constructor properties. The requirement of $\langle c, s\rangle$-*closedness* in (While) is in order to achieve total correctness. A predicate p is $\langle c, s\rangle$-closed if every sequence of probabilistic states satisfying p and obtained after an increasing number of iterations of the conditional **if** c **then** s **else skip fi** converge in the limit to a state also satisfying p. The verification of this condition is automatic for terminating loops. The only unbounded loop is in $Uniform$. We ommit the proof here but it can be found in [22].

4.2. Correctness Proof

As before, we divide the proof in $FactorTwos$, $Uniform$ and $Witness$. We then combine them in two different cases depending on n being prime or not. One minor difference is the introduction of constants N and T that contain the input values of variables n and t. They are required because of their use outside $\mathbb{P}$-predicates (see the example above). Since in particular, $n = N$ remains invariant along the program, we work under the assumption that $\mathbb{P}(n \neq N) = 0$ holds along the program and we include it as a tautology according to our convenience. (This is in order to avoid the assumption as a precondition and carry the statement along all proof outlines.)

$FactorTwos$**.** Contrarily to previous section, we report the proof of this section of the algorithm since its connection to Hoare logic is not immediate. Besides, it provides a smoother introduction to the manipulation of the pH logic. Define the loop invariant $Inv : \mathbb{P}(n{-}1 = 2^r s) = 1$. Notice that it is equivalent to $(\exists r_1, r_2 : q)$, where

$$q : r_1 + r_2 = 1 \wedge \mathbb{P}(n{-}1 = 2^r s \wedge s\ mod\ 2 = 0) \geq r_1 \wedge \mathbb{P}(n{-}1 = 2^r s \wedge s\ mod\ 2 \neq 0) \geq r_2$$

In order to carry out the verification, we use *proof outlines* as it is usually done in Hoare logic. A particular difference arises while proving loops since Inv needs to be prove invariant in a conditional construction rather than in the loop body. We remark that there is no real calculus reasoning behind probabilistic predicates, except for the usual predicate logic and arithmetics. We then have to resort to semantic definitions in order to prove implications. The proof outline for $FactorTwos$ is as follows.

```
 { $\mathbb{P}(true) = 1$ }
 $r := 0;\ s := n{-}1;$
 { $Inv$ }
 { $q$ }            /* existential is removed using rule (Exists) */
 do $s\ mod\ 2 = 0 \rightarrow$
       { $(s\ mod\ 2 = 0)?q$ }
    ⇒ { $r_1 + r_2 = 1 \wedge \mathbb{P}(n{-}1 = 2^{r+1}(s\ div\ 2)) \geq r_1$ }
       $s := s\ div\ 2; r := r + 1$
       { $r_1 + r_2 = 1 \wedge \mathbb{P}(n{-}1 = 2^r s) \geq r_1$ }
       ----------------------------------------------------------------
       { $(s\ mod\ 2 \neq 0)?q$ } skip { $(s\ mod\ 2 \neq 0)?q$ } ⇒ { $\mathbb{P}(n{-}1 = 2^r s) \geq r_2$ }
       ----------------------------------------------------------------
       { $(r_1 + r_2 = 1 \wedge \mathbb{P}(n{-}1 = 2^r s) \geq r_1)\ +\ (\mathbb{P}(n{-}1 = 2^r s) \geq r_2)$ } ⇒ { $Inv$ }
 od
 { $Inv\ \wedge\ \mathbb{P}(s\ mod\ 2 = 0) = 0$ }
⇒ { $\mathbb{P}(n{-}1 = 2^r s \wedge \mathsf{odd}.s) = 1$ }
```

Note the proof scheme for the loop. It is divided in three sections. The first one corresponds to the positive branch of the **if** in the premise of rule (While) (notice the positive cut in precondition $(s\ mod\ 2 = 0)?q$). The second one represents the negative branch where only **skip** is performed (notice the negative cut in precondition $(s\ mod\ 2\ = 0)?q$). Finally, the third section shows how both branches are combined using probabilistic predicate operator $+$ to finally obtain the invariant Inv. We also remark that rule (Exists) seems to be applied earlier than strictly necessary. Normally this rule is only applied in the **if** construction in the premise of (While). Since the **if** is omitted here, we include this derivation in the outlines before the **do**. We do so in general.

$Uniform$**.** For the inner loop in $Uniform'$, define invariant $Inv : (\exists i : q)$ where

$$q : \left(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N - \mathsf{lg2}.i} : \mathbb{P}(a = j \wedge k = i) \geq 2^{-\mathsf{lg2}.N + \mathsf{lg2}.i}\right)$$

It states that at iteration $\mathsf{lg2}.N - \mathsf{lg2}.k$, a is uniformly distributed in the interval $[0, 2^{\mathsf{lg2}.N-\mathsf{lg2}.k})$.

```
     $\{ \mathbb{P}(true) = 1 \}$
  $\Rightarrow$ $\{ \mathbb{P}(n = N) = 1 \}$          /* $\mathbb{P}(n = N) = 1$ is assumed to hold */
     $a := 0;\ k := n;$
     $\{ \mathbb{P}(a = 0 \wedge k = N) = 1 \}$          /* by (Assign) */
  $\Rightarrow$ $\{ Inv \}$
     $\{ q \}$          /* by (Exists) */
     do $k \neq 0 \ \rightarrow$
          $\{ (k \neq 0)?q \}$
       $\Rightarrow$ $\{ \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N-\mathsf{lg2}.(i\ div\ 2)-1} : \mathbb{P}(a = j \wedge k\ div\ 2 = i\ div\ 2) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.(i\ div\ 2)+1}\big) \}$
          $a := 2a \ \oplus_{1/2}\ a := 2a + 1;$
          $\{ \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N-\mathsf{lg2}.(i\ div\ 2)} \wedge \mathsf{even}.j\ : \mathbb{P}(a = j \wedge k\ div\ 2 = i\ div\ 2) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.(i\ div\ 2)+1}\big)$
        $\oplus_{1/2} \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N-\mathsf{lg2}.(i\ div\ 2)} \wedge \mathsf{odd}.j\ : \mathbb{P}(a = j \wedge k\ div\ 2 = i\ div\ 2) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.(i\ div\ 2)+1}\big) \}$
       $\Rightarrow$ $\{ \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N-\mathsf{lg2}.(i\ div\ 2)} : \mathbb{P}(a = j \wedge k\ div\ 2 = i\ div\ 2) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.(i\ div\ 2)}\big) \}$
          $k := k\ div\ 2$
          $\{ \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N-\mathsf{lg2}.(i\ div\ 2)} : \mathbb{P}(a = j \wedge k = i\ div\ 2) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.(i\ div\ 2)}\big) \}$
       $\Rightarrow$ $\{ q[i/i\ div\ 2] \}$
          ----------------------------------------------------------------------------------
          $\{\neg(k \neq 0)?q \}$ skip $\{\neg(k \neq 0)?q \} \Rightarrow \{ true \}$
          ----------------------------------------------------------------------------------
          $\{ q[i/i\ div\ 2] + true \} \ \Rightarrow\ \{ Inv \}$
     od
     $\{ Inv \wedge \mathbb{P}(k \neq 0) = 0\}$
  $\Rightarrow$ $\{ \big(\forall j : 0 \leq j < 2^{\mathsf{lg2}.N} : \mathbb{P}(a = j) = 2^{-\mathsf{lg2}.N}\big) \}$
```

In the first section of the loop (positive branch), we use $i \neq 0 \Rightarrow \mathsf{lg2}.i = \mathsf{lg2}.(i\ div\ 2) + 1$, and rules (Prob) and (Assign). The last implication follows by definition of q. The second section is straightforward use of (Skip) and (Cons). The last section containing the $+$ combination makes use of (Cons) and the fact that $\mathbb{P}(dp) \geq r + true \Rightarrow \mathbb{P}(dp) \geq r$ to show that the invariant is maintained. The last implication after the **do** construction is a consequence of the observation that if one of the inequalities $\mathbb{P}(a = j \wedge k = i) \geq 2^{-\mathsf{lg2}.N+\mathsf{lg2}.i}$ were strict (notice that at this point $i = 0$ since $\mathbb{P}(k \neq 0) = 0$ and hence $\mathbb{P}(a = j) \geq 2^{-\mathsf{lg2}.N}$), the contradiction $\mathbb{P}(true) = \sum_j \mathbb{P}(a = j) > 1$ is obtained.

To prove the main loop $Uniform$ we introduce an auxiliary variable z that counts the number of iterations of $Uniform$; recall that it could iterate an unbounded number of times. The invariant of the main loop in $Uniform$ is defined as follows:

$$\begin{array}{rcl} Inv & : & (\exists i : q) \vee (\forall j : 0 < j : p) \\ q & : & \mathbb{P}(z = i \wedge \neg(0 < a < n)) = (1-\rho)^i \ \wedge \ (\forall j : 1 \leq j \leq i : p) \\ p & : & \big(\forall k : 0 < k < N : \mathbb{P}(z = j \wedge a = k) = (1-\rho)^{j-1} * \rho/(N-1)\big) \end{array}$$

where $\rho = (N-1)2^{-\mathsf{lg2}.N}$. Predicate p states the probability of terminating in the jth iteration (notice that it is independent of the value of a). Predicate q talks about the probability of terminating within i iterations (second conjunct) and the probability that a is generated outside the range $[1, N-1]$ in the ith iteration. Therefore, the left disjunct in Inv refers to the probability of either terminating with an appropriate value in a and the probability that this does not occur within some i iterations. The right disjunct states that for each iteration j the loop terminates with some probability, and this condition is needed in order to show termination.

In the next proof outline, the vertical line corresponds to the application of rule (Or) where the left and right proof outlines correspond to the two triples on the premise of the rule.

```
$\{ \mathbb{P}(true) = 1 \}$
$z := 0;\ a := n;$
$\{ Inv \}$
----------------------------------------------------------------
$\{ (\exists i : q) \}$
$\{ q \}$
do $\neg(0 < a < n) \rightarrow$
      $\{\neg(0 < a < n)?q \}$
   $\Rightarrow$ $\{ \mathbb{P}(z = i \wedge \neg(0 < a < n)) = (1-\rho)^i \}$
   $\Rightarrow$ $\{ (1-\rho)^i \cdot [\mathbb{P}(z = i \wedge \neg(0 < a < n)) = 1] \}$
   $\Rightarrow$ $\{ (1-\rho)^i \cdot [\mathbb{P}(true) = 1 \wedge \mathbb{P}(z = i) = 1]\}$
      $Uniform'$;
      $\{ (1-\rho)^i \cdot [\big(\forall k : 0 < k < 2^{\mathsf{lg2}.N} : \mathbb{P}(a = k) = 2^{-\mathsf{lg2}.N}\big) \wedge$
                $\mathbb{P}(z = i) = 1]\}$
      /* this last step holds by $(\mathrm{Lin}\ \cdot)$, orthogonality on $\mathbb{P}(z = i) = 1$, and */
      /* the result above on $Uniform'$ */
   $\Rightarrow$ $\{ (1-\rho)^i \cdot [\mathbb{P}(z = i \wedge \neg(0 < a < n)) = (1-\rho) \wedge$
                $(\forall k : 0 < k < N : \mathbb{P}(z = i \wedge a = k) = \rho/(N-1))] \}$
   $\Rightarrow$ $\{ \mathbb{P}(z+1 = i+1 \wedge \neg(0 < a < n)) = (1-\rho)^{i+1} \wedge$
       $\big(\forall k : 0 < k < N : \mathbb{P}(z+1 = i+1 \wedge a = k) = (1-\rho)^i * \rho/(N-1)\big) \}$
      $z := z + 1$
      $\{ \mathbb{P}(z = i+1 \wedge \neg(0 < a < n)) = (1-\rho)^{i+1} \wedge p[j/i+1] \}$
      ----------------------------------------------------------
      $\{ (0 < a < n)?q \}$ skip $\{ (0 < a < n)?q \} \Rightarrow \{ (\forall j : 1 \le j \le i : p) \}$
      ----------------------------------------------------------
      $\{\big( \mathbb{P}(z = i+1 \wedge \neg(0 < a < n)) = (1-\rho)^{i+1} \wedge p[j/i+1] \big)$
       $+ (\forall j : 1 \le j \le i : p) \}$
   $\Rightarrow$ $\{ q[i/i+1] \} \Rightarrow \{ Inv \}$
od
$\{ Inv \wedge \mathbb{P}(\neg(0 < a < n)) = 0 \}$

$\{ (\forall j : 0 < j : p) \}$
do $\neg(0 < a < n) \rightarrow$
      $\{\neg(0 < a < n)?(\forall j : 0 < j : p) \}$
   $\Rightarrow$ $\{\mathbb{P}(true) = 0\}$
      $Uniform'$;
      $z := z + 1$
      $\{\mathbb{P}(true) = 0\}$
      ----------------------------------------------------------
      $\{ (0 < a < n)?(\forall j : 0 < j : p) \}$
   $\Rightarrow$ $\{ (\forall j : 0 < j : p) \}$
      skip
      $\{ (\forall j : 0 < j : p) \}$
      ----------------------------------------------------------
      $\{ \mathbb{P}(true) = 0\ + (\forall j : 0 < j : p) \}$
   $\Rightarrow \{ (\forall j : 0 < j : p) \} \Rightarrow \{ Inv \}$
od
$\{ Inv \wedge \mathbb{P}(\neg(0 < a < n)) = 0 \}$
----------------------------------------------------------------
$\Rightarrow \{ (\forall k : 0 < k < N : \mathbb{P}(a = k) = 1/(N-1)) \}$
```

The implication at the very end is obtained as a result of the same series used for the $pGCL$ verification.

$Witness$**.** Like for $pGCL$, this proof is long, non-probabilistic and mostly straightforward. We omit it here and refer to [22] for further details. We only report the loop invariant and the triple:

$$Inv\ :\ \mathbb{P}\big(a' = a^{2^y s}\ mod\ n \wedge witness = \mathsf{WITNESS}'.n.a.y\big) = 1$$
$$\{ \mathbb{P}(true) = 1 \}\ Witness\ \{ \mathbb{P}(witness = \mathsf{WITNESS}.n.a) = 1 \}$$

One iteration. As for $pGCL$, we split the proof in two cases. For the case $\neg\mathsf{PRIME}.n$ we calculate:

```
   $\{ \mathbb{P}(\neg\mathsf{PRIME}.n) = 1 \}$
$\Rightarrow$ $\{ \mathbb{P}(true) = 1 \wedge \mathbb{P}(\neg\mathsf{PRIME}.n) = 1 \}$
   $Uniform$;
   $\{ (\forall j : 0 < j < N : \mathbb{P}(a = j) = 1/(N-1)) \wedge \mathbb{P}(\neg\mathsf{PRIME}.n) = 1 \}$     /* by orthogonality and result on $Uniform$ */
$\Rightarrow$ $\{ \mathbb{P}(true) = 1 \wedge \mathbb{P}(\mathsf{WITNESS}.n.a) \ge 1/2 \}$
   $Witness$
   $\{ \mathbb{P}(witness = \mathsf{WITNESS}.n.a) = 1 \wedge \mathbb{P}(\mathsf{WITNESS}.n.a) \ge 1/2 \}$     /* by orthogonality and result on $Witness$ */
$\Rightarrow \{ \mathbb{P}(\neg witness) \le 1/2 \}$                                                                      (8)
```

In the second implication we use (4) (see Sec. 3.2), which says that at least half of the bases in the range $[1, N-1]$ are witnesses when n is not prime.

The proof for $\mathsf{PRIME}.n$ is simpler:

```
   $\{ \mathbb{P}(\mathsf{PRIME}.n) = 1 \}$
   $Uniform$;
   $\{ \mathbb{P}(\mathsf{PRIME}.n) = 1 \}$                                            /* by orthogonality */
   $Witness$
   $\{ \mathbb{P}(\mathsf{PRIME}.n) = 1 \wedge \mathbb{P}(witness = \mathsf{WITNESS}.n.a) = 1 \}$   /* by orthogonality and result on $Witness$ */
$\Rightarrow$ $\{ \mathbb{P}(\neg witness) = 1 \}$                                            /* by (3) */
```

Outer loop. If $\neg\mathsf{PRIME}.n$, we define $Inv\ :\ (\exists i\ :\ \mathbb{P}(prime)\ \le\ 2^{-T+i} \wedge \mathbb{P}(t\ =\ i \wedge \neg\mathsf{PRIME}.n)\ =\ 1)$ which shows that the probability of the algorithm to err decreases exponentially in each iteration. Notice that Inv is equivalent to $(\exists i : q) \vee p$ where

$$\begin{array}{lcl} p & : & \mathbb{P}(prime) \leq 2^{-T} \wedge \mathbb{P}(t = 0 \wedge \neg \mathsf{PRIME}.n) = 1 \\ q & : & \mathbb{P}(prime) \leq 2^{-T+i} \wedge \mathbb{P}(0 < t \wedge t = i \wedge \neg \mathsf{PRIME}.n) = 1 \end{array}$$

The following result, which will be of use below, follows by orthogonality, (Assign) and some simple calculations:

$$\begin{array}{l} \{\ \mathbb{P}(prime) = 1 \wedge \mathbb{P}(\neg \mathsf{PRIME}.n) = 1\ \} \\ Uniform;\ Witness;\ prime := prime \wedge \neg witness \\ \{\ \mathbb{P}(prime) \leq 1/2\ \} \end{array} \tag{9}$$

In the calculation below, we apply rule (Or) and use the same format as before.

```
   { $\mathbb{P}(\neg \mathsf{PRIME}.n) = 1 \wedge \mathbb{P}(T = t) = 1$ }
   $prime := true$;
   { $\mathbb{P}(\neg \mathsf{PRIME}.n) = 1 \wedge \mathbb{P}(T = t) = 1 \wedge \mathbb{P}(prime) = 1$ }
⇒ { $Inv$ }
----------------------------------------------------------------------
   { $(\exists i : q)$ }
   { $q$ }          /* by (Exists) */
   do $t \neq 0 \rightarrow$
         { $(t \neq 0)?q$ }
       ⇒ { $\exists r : \mathbb{P}(prime) = r \wedge r \leq 2^{-T+i} \wedge \mathbb{P}(t = i \wedge \neg \mathsf{PRIME}.n) = 1$ }
         { $\mathbb{P}(prime) = r \wedge r \leq 2^{-T+i} \wedge \mathbb{P}(t = i \wedge \neg \mathsf{PRIME}.n) = 1$ }
         /* last step by (Exists) */
       ⇒ { $r \cdot [\mathbb{P}(\neg \mathsf{PRIME}.n) = 1] \wedge r \leq 2^{-T+i} \wedge \mathbb{P}(t = i \wedge \neg \mathsf{PRIME}.n) = 1$ }
         $Uniform;\ Witness;\ prime := prime \wedge \neg witness$;
         { $r \cdot [\mathbb{P}(prime) \leq 1/2] \wedge r \leq 2^{-T+i} \wedge \mathbb{P}(t = i \wedge \neg \mathsf{PRIME}.n) = 1$ }
         /* last step holds by (Lin ·), orthogonality and (9) */
       ⇒ { $\mathbb{P}(prime) \leq 2^{-T+i-1} \wedge \mathbb{P}(t - 1 = i - 1 \wedge \neg \mathsf{PRIME}.n) = 1$ }
         $t := t - 1$
         { $\mathbb{P}(prime) \leq 2^{-T+i-1} \wedge \mathbb{P}(t = i - 1 \wedge \neg \mathsf{PRIME}.n) = 1$ }
         ---------------------------------------------------------
         { $\neg(t \neq 0)?q$ } skip { $\neg(t \neq 0)?q$ } ⇒ { $\mathbb{P}(true) = 0$ }
         ---------------------------------------------------------
         { $\big(\mathbb{P}(prime) \leq 2^{-T+i-1} \wedge \mathbb{P}(t = i - 1 \wedge \neg \mathsf{PRIME}.n) = 1\big)$
          $+\ (\mathbb{P}(true) = 0)$ }
       ⇒ { $q[i/i-1]$ } ⇒ { $Inv$ }
   od
   { $Inv \wedge \mathbb{P}(t \neq 0) = 0$ }

   { $p$ }
   do $t \neq 0 \rightarrow$
         { $(t \neq 0)?p$ }
       ⇒ { $\mathbb{P}(true) = 0$ }
         $Uniform$;
         $Witness$;
         $prime := prime \wedge \neg witness$;
         $t := t - 1$
         { $\mathbb{P}(true) = 0$ }
         ---------------------------------
         { $\neg(t \neq 0)?p$ }
         skip
         { $\neg(t \neq 0)?p$ } ⇒ { $p$ }
         ---------------------------------
         { $\mathbb{P}(true) = 0 + p$ }
       ⇒ { $p$ } ⇒ { $Inv$ }
   od
   { $Inv \wedge \mathbb{P}(t \neq 0) = 0$ }
----------------------------------------------------------------------
⇒ { $\mathbb{P}(\neg prime) \geq 1 - 2^{-T}$ }
```

If n is prime, take $Inv : \mathbb{P}(\mathsf{PRIME}.n) = 1 \wedge \mathbb{P}(prime) = 1$. Notice that $\{Inv\}\ Uniform;\ Witness\ \{Inv \wedge \mathbb{P}(\neg witness) = 1\}\ prime := prime \wedge \neg witness\ \{Inv\}$. By observing that $Inv \equiv ((t \neq 0) \wedge Inv) \vee ((t = 0) \wedge Inv)$ and applying rule (Or), it is straightforward to calculate the following.

$$\begin{array}{l} \{\ \mathbb{P}(\mathsf{PRIME}.n) = 1\ \} \\ prime := true; \\ \{\ Inv\ \} \\ \textbf{do}\ t \neq 0 \rightarrow Uniform;\ Witness;\ prime := prime \wedge \neg witness;\ t := t - 1\ \textbf{od} \\ \{\ Inv \wedge \neg(t \neq 0)\ \} \Rightarrow \{\ \mathbb{P}(prime) = 1\ \} \end{array}$$

5. Conclusions

In the following we report our conclusions on the use of both verification techniques. We compare them focusing their expressiveness and complexity on formal manipulation. We also hint some improvements and further research directions.

Expressiveness. We divide this part according to the underlying logic, the programming language, and the full calculus. The logic behind $pGCL$ is the usual predicate logic but reinterpreted in the real interval $[0, 1]$. Such interpretation should be understood as the expected probability that the property holds. Instead pH provides a richer language for the logic where formulas are allowed to talk about the probability of events (described as plain predicate logic) but still interpreted in the usual boolean lattice. Therefore, it allows, for instance, to compare probabilities of different events as in $\mathbb{P}(x > 4) < \mathbb{P}(y = z)$.

With respect to the programming language under the calculus, there are some characteristics that, though not evident in our article, they are worth to mention. First, the full $pGCL$ calculus is build based on Jones [12] and Kozen [14] calculi by extending it with a non-deterministic operation. There is also a version of pH that includes nondeterminism, but it does not allow for loops [7]. Second, $pGCL$ allows for parameterized probabilistic operation (i.e., the subindex ρ in $\oplus_\rho$ could be any expression containing program variables) while pH only allows for constant values.

If restricted to deterministic language, the proof system for pH is richer than the weakest pre-expectation calculus. $pGCL$ formula and pH triple

$$\rho * [p] \Rrightarrow wp.s.[q] \qquad \text{and} \qquad \{\mathbb{P}(p) = 1\}\ s\ \{\mathbb{P}(q) \geq \rho\} \tag{10}$$

are equivalent. Consequently, pH allows to verify richer properties in only one proof. A particular example arises in the proof of one iteration of the main loop of the Miller-Rabin for the case $\neg\mathsf{PRIME}.n$. $pGCL$ requires two different proof to asses that variable $witness$ is valid with probability larger than 1/2 (equation (5)) or that it is false with probability smaller than 1/2 (equation (6)). (Notice that in general is not true that $wp.s.[p] \equiv 1 - wp.s.[\neg p]$ since s may not terminate with certain probability). Instead, pH allows to deduce one from the other in only one step (see (8)).

It is the view of the authors that a pH triple is usually easier to understand than a $pGCL$ formula for the following reasons. First, probabilistic predicates in pH are interpreted as true or false, while in $pGCL$ the interpretation is on the real line. Second, in a pGCL expression $P \Rrightarrow wp.s.Q$ predicates P and Q are *not* the pre- and post-condition of program s. This can be observed in (10) where the value ρ, representing the probability of the postcondition, appears on the lefthand side (i.e., in P). This has been particularly disturbing at the moment of finding loop invariants in $pGCL$. They usually were obtained by fine tuning rather than intuition. (Noticeably, the intuition reported above for loop invariant I in $Uniform'$ wrongly –but intentionally– binds operation $*$ to an equality when we say that the probability "is $2^{-\mathsf{lg2}.k}$".) On the contrary, the notion of invariant in pH is fully compatible with that in Hoare logic since in both cases, assertion $\{P\}$ in a proof outline completely describe the set of states that makes P valid at that point of the program.

Complexity of the proofs. $pGCL$ provides a mature and simple calculus. The simplicity arises for a clear separation between predicate logic (appearing only between square brackets) and functions (which are the interpretation of such predicates). The manipulation of both types of expressions are well understood. Besides, the years of development and study of $pGCL$ provided a significant number of properties and theorems that facilitates its manipulation (see Section 3.1). It is our experience that once familiarized with the interpretation of the logic, calculations with $pGCL$ are fairly easy.

On the other side, pH has been much harder to manipulate for precisely the opposite reasons. Both the significant expressibility and the lack of development behind this calculus obliged us to go down to basic semantic manipulations. On the one hand, it is the lack of an axiomatization for the new operations of the probabilistic logic, namely summation ($P + Q$), scalar multiplication ($\rho \cdot P$), and "cut" ($c?P$). On the other hand, proving loop termination demands to check for $\langle c, s\rangle$-closedness, a pure semantic notion that requires fairly complex mathematic manipulation. Compare this to $pGCL$ where we only used existing theorems and lemmas.

Therefore, there is still much to develop on pH. On this paper, in particular, we improved the proof outline style (compared to [7]) making proofs more self contained (rather than splitted in several lemmas). Moreover, our proofs on loops (with the exception of $FactorTwos$ and $Uniform$) follows the same pattern: split the invariant in two disjoint disjuncts, one holding only if the guard of the loop is true, and the other holding only if the guard is false, and apply rule (Or) on this disjunct. (This proof process was introduced in [7].) This suggest the following rule

$$\frac{\begin{array}{c}(p \equiv q_1 \vee q_2) \quad \wedge \quad ((\neg c?q_1) \equiv (\mathbb{P}(true) = 0)) \quad \wedge \quad ((c?q_2) \equiv (\mathbb{P}(true) = 0)) \\ \{c?q_1\}\ s\ \{p\} \qquad (\neg c?q_2) \Rightarrow p \qquad p \text{ is } \langle c, s\rangle\text{-closed}\end{array}}{\{p\}\ \mathbf{do}\ c \rightarrow s\ \mathbf{od}\{p \wedge \mathbb{P}(c) = 0\}} \quad (\text{While}')$$

where the predicate in the top defines the splitting of invariant p, $\{c?q_1\}\ s\ \{p\}$ is a simplification of the lefthand side of rule (Or), and $(\neg c?q_2) \Rightarrow p$ is a simplification of the righthand side.

Concurrent probabilistic programs. This comparison was motivated as a way to understand assertional calculi for probabilistic programs with a long-term aim of developing an assertional calculus for concurrency. The weakest pre-expectation calculus includes a non-deterministic construction which serves well in order to verify abstract models of concurrent programs [4] using refinement techniques. Nevertheless, a compositional technique, such as that of Owicki-Gries for non-probabilistic program [19], provide a direct verification of concurrent systems. In this direction pH seems more suitable because of its potential of proof outlining (which is central in Owicki-Gries logic). Alternatively, one can think of Jones' work [12] which also presents Hoare like triples (and hence potential for proof outlines), but like $pGCL$, assertions are interpreted in the reals.

Thanks

We want to thank especially to Jerry den Hartog and Carroll Morgan for their helpful comments.

References

[1] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. Report, Dep. of Computer Science and Engineering, Indian Institute of Technology Kanpur, August 2002.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, Cambridge, Massachusetts, 1990.

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

[4] C. Fidge and C. Shankland. But what if i don't wait forever? *Formal Aspects of Computing*, 14(3):281–294, April 2003.

[5] R. Gupta, S.A. Smolka, and S. Bhaskar. On randomization in sequential and distributed algorithms. *ACM Computing Surveys*, 26(1):7–86, 1994.

[6] J.I. den Hartog. Verifying probabilistic programs using a hoare like logic. In P.S. Thiagarajan and R. Yap, editors, *LNCS 1742 (ASIAN'99)*, pages 113–125. Springer, 1999.

[7] J.I. den Hartog. *Probabilistic Extensions of Semantical Models*. PhD thesis, Vrije Universiteit Amsterdam, October 2002.

[8] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of ACM*, 12:576–580, 1969.

[9] J. Hurd. *Formal Verification of Probabilistic Algorithms*. PhD thesis, Univ. of Cambridge, 2002.

[10] J. Hurd. Verification of the Miller-Rabin probabilistic primality test. *Journal of Logic and Algebraic Programming*, 50(1–2):3–21, May–August 2003.

[11] J. Hurd, A. McIver, and C. Morgan. Probabilistic guarded commands mechanized in hol. *2nd Workshop on Quantitative Aspects of Programming Languages*, 2004.

[12] C. Jones. *Probabilistic Non-determinism.* PhD thesis, University of Edinburgh, 1990.

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

[14] D. Kozen. A probabilistic pdl. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 291–297. ACM Press, 1983.

[15] C. Morgan. Proof rules for probabilistic loops. In He Jifeng, J. Cooke, and P. Wallis, editors, *BCS-FACS 7th Refinement Workshop*. Springer, 1996.

[16] C. Morgan and A. McIver. pGCL: Formal reasoning for random algorithms. *South African Computer Journal*, 22:14–27, March 1999.

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

[18] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[19] Susan S. Owicki. Axiomatic proof techniques for parallel programs. Technical Report TR75-251, Cornell University, Computer Science Department, jul 1975.

[20] M.O. Rabin. *Probabilistic Algorithms*, pages 21–39. Academic Press, NY, 1976.

[21] K. Seidel, C. Morgan, and A. McIver. Probabilistic imperative programming: a rigorous approach. In L. Groves and S. Reeves, editors, *Formal Methods Pacific '97*. Springer, 1997.

[22] M. Vasquez. Verificación Comparativa del Algoritmo de Miller-Rabin. Master's thesis, Faculty of Mathematics, Astronomy and Physics, National University of Córdoba, May 2004.