Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía y Física

Una verificación comparativa del algoritmo de Miller-Rabin.

Miguel Vasquez [1]

2004

[1] e-mail: mvasquez@hal.famaf.unc.edu.ar

Índice General

Capítulo 1

Introducción

La probabilidad entra en escena para modelar ciertos procesos que tienen un comportamiento inherentemente probabilístico o no del todo conocido. Los algoritmos probabilísticos tienen como objetivo darle un mejor trato a problemas que no pueden ser resueltos eficientemente, o que no pueden ser resueltos del todo de manera estandard. Los problemas complejos requieren a menudo de procesos matemáticos que requieren mucho poder de cómputo; con lo cual usar este tipo de algoritmos resulta una buena opción para reducir la complejidad del tiempo de cálculo. En [9] Gupta explica las ventajas de los algoritmos probabilísticos que se están volviendo cada vez más populares; ello se debe a que casi siempre resultan más fácil de implementar y entender que su versión estandard. Esta simpleza ha sido un factor determinante para la aceptación de los mismos en la comunidad del software; en la que se le da mucha importancia al costo y complejidad. Sin embargo, como contrapunto está el hecho de que se sacrifica la noción de corrección absoluta del programa, por una noción de corrección con probabilidad mayor o igual a $1 - \epsilon$. La belleza de estos algoritmos radica en que ejecutando sólo algunas iteraciones se obtiene un grado alto de confibialidad (ϵ pequeño) en el resultado. Lo cual suele ser a menudo mucho mas rápido que ejecutar un complicado algoritmo estandard una vez.

Muchos sistemas de criptografia de clave pública como el *RSA*, basan su seguridad en la dificultad de factorizar grandes numeros. De allí nace la importancia de generar numeros primos de 100 o más dígitos. Verificar si un entero es primo, es un problema clasico de teoría de números. Introduciendo probabilidad puede resolverse en tiempo polinomial. En este trabajo se estudiará el test de primalidad de *Miller-Rabin*; el cual pertenece al tipo de algoritmos *Monte Carlo* que son rápidos y probablemente correctos. Usando una notación general el algoritmo satisface dos resultados:

$$\begin{array}{rcll} \mathsf{prime}.N & \Longrightarrow & Prob\big[\ MillerRabin(N,T)\ \big] & = 1 \\ \neg\mathsf{prime}.N & \Longrightarrow & Prob\big[\neg MillerRabin(N,T)\ \big] & \geq 1 - 2^{-T} \end{array}$$

donde T representa la cantidad de iteraciones. Es decir que el error 2^{-T} puede hacerse exponencialmente tan pequeño como se desee iterando varias veces. El algoritmo utiliza una técnica probabilística que se denomina *búsqueda aleatoria* [9]; que consiste en buscar en un espacio grande por algún elemento que tenga la propiedad deseada. Si la propiedad en cuestión es fácilmente verificada y los elementos que la poseen son abundantes, entonces la búsqueda aleatoria puede ser muy efectiva. Un requisito implícito

es que la búsqueda sea mas o menos uniforme del espacio en consideración. En el caso de Miller-Rabin, siendo N el número sobre el cual se está chequeando primalidad, entonces el espacio de búsqueda es el conjunto $\{2, \ldots, N-1\}$. Si un elemento de ese conjunto satisface cierta propiedad a verificar, entonces es compuesto. En [12] Miller probó que existen al menos $\frac{N-1}{2}$ elementos con cierta propiedad (presentada en el próximo capítulo) que atestigua que N no es primo.

La presencia de elementos probabilísticos en los programas hace que el entendimiento operacional y testing se conviertan precarios. Entonces herramientas adicionales, tales como las técnicas de verificación formal, se convierten fundamentales para el diseño de programas probabilísticos. Hurd [2] hizo una verificación del test de Miller-Rabin en un marco funcional; aquí se hará en uno imperativo. Se verificará el algoritmo usando dos lógicas totalmente distintas que son extensiones probabilísticas de la lógica de *Hoare*. A los comandos guardados usuales de Dijkstra [1], se incorpora un nuevo constructor que se denomina *selección probabilística*. Supongamos que ρ es una constante en el rango $(0, 1)$ entonces el programa $S \oplus_\rho T$ se interpreta operacionalmente asi: con probabilidad ρ se ejecuta el programa S y con probabilidad $1 - \rho$ se ejecuta el programa T. Antes de continuar vamos a diferenciar varios tipos de programas:

Término	Significado
estandard	no incluye ningún operador $\oplus_\rho$ o $\sqcap$
probabilístico	incluye al menos un operador $\oplus_\rho$
determinístico	no incluye ningún un operador $\sqcap$
no-determinístico	incluye al menos un operador $\sqcap$

El programa *no-determinístico* $S \sqcap T$ representa la ignorancia o abstracción total de cual de los dos programas será ejecutado. De todos modos aquí sólo se trabajará con programas determinísticos.

La primer lógica es la extensión de *Hoare* creada por *Hartog* [4]. Aqui se propone un sistema de prueba completo y correcto para razonar sobre programas que actúan probabilísticamente. Este sistema se denomina pH, y con el se puede probar la validez de triplas de *Hoare* $\{p\}s\{q\}$; donde p es una precondición y q es una postcondición del programa probabilístico s. Se introducen *predicados probabilísticos* que permiten expresar probabilidades sobre predicados determinísticos o decir que una variable de programa tiene cierta distribución de probabilidades.

La segunda extensión de *Hoare* se denomina *pCGL*; está desarrollada entre otros por *Morgan* [5], [6], [7], [8]. En esta lógica se trata la verificación de los programas probabilísticos a nivel de *weakest preconditions*. Un predicado estandard P se traslada a la aritmética escribiendolo como el predicado probabilístico $[P]$, una expresión que vale 1 cuando se satisface P y 0 en caso contrario. El trasformador de predicados probabilísticos $wp.S$ facilita el razonamiento de programas imperativos que contengan alguna selección probabilística. No sólo se puede determinar si un programa establece cierto resultado, sino que además con que probabilidad lo hace. Por otra parte se generalizan las nociones de *invarianza* y *variante* agregando argumentos probabilísticos para la terminación de un programa.

En el capítulo 2 se presenta con detalle el algoritmo de Miller-Rabin y dos propiedades matemáticas fundamentales para la verificación con cada una de las lógicas. En el capítulo 3 da una introducción de la lógica de *Hartog*; y la correspondiente verificación del algoritmo. En el capítulo 4 se introduce la lógica de *Morgan* y posteriormente la verificación de Miller-Rabin. El capítulo 5 establece una comparación

de ambas lógicas basada en las dos verificaciones, resaltando los puntos a favor y en contra de cada lógica. En el final se incluyen dos apéndices con propiedades y lemas auxiliares que fueron utilizados en la verificación del algoritmo.

Capítulo 2

Algoritmo de *MillerRabin*

2.1 Algoritmo

Aquí se presentará una versión con algunas variaciones del original que *Miller* publicó en [13]. Los cambios no son trascendentes, pero son convenientes para que las verificaciones resulten más entendibles. Por ejemplo hay variables como b, a de las cuales se podría prescindir; pero que sin embargo se las usa para razonar y verificar mejor el algoritmo. El entero sobre el cual se está testeando primalidad satisface $N \geq 3$. De todos modos ya sabemos que 1 no es primo y 2 sí lo es. Por completitud se podría anidar el siguiente algoritmo dentro de un `if` para contemplar los casos: $n = 1, 2$.

con N, T : Int;
var $x, y, a, a', n, r, s, k, m, b$: Int;
var $prime, witness$: $Bool$;

$n = N$
$FactorTwos$;
$x := 0$;
$prime := true$;
`while` $(x < T)$ `do`
  $Uniform_a[2, n-1]$; $WitnessTail$; } $MillerRabin1$
  $prime := prime \wedge \neg witness$;
  $x := x + 1$
`od`

} $MillerRabin$

Seguidamente se explicará en que consiste cada una de las partes que compone el algoritmo principal.

Programa $FactorTwos$ **:**

$r := 0;$
$s := n - 1;$
`while` $(s$ `mod` $2 = 0)$ `do`
    $s := s$ `div` $2;$
    $r := r + 1$
`od`

Este es simplemente un programa auxiliar que encuentra naturales r, s tales que s es impar y $2^r s = n-1$.

Programa $Uniform_a[2, n-1]$ **:**

$m := n;$
`while` $\neg(2 \leq m < n)$ `do`
    $UnifPow2_n$
`od` ;
$a := m$

Este programa está inspirado en la versión aplicativa prob_uniform definida por Hurd [3]. Se usa un ciclo para ejecutar repetidamente $UnifPow2_n$, deteniéndose cuando $UnifPow2_n$ retorna en m un numero en el rango correcto $[2, n)$. Notar que $Uniform_a[2, n-1]$ solo está definido para $n \geq 3$, ya que la distribución Uniform$[2, 2)$ es un concepto matemático mal formado.

Programa $UnifPow2_n$ **:**

$k := n;$
$m := 0;$
`while` $(k \neq 0)$ `do`
    $k := k$ `div` $2;$
    $b := 0 \ \oplus_{\frac{1}{2}} \ b := 1;$
    $m := 2m + b$
`od`

Este programa es una adaptación imperativa del algoritmo prob_unif definido en un marco funcional por Hurd [3]. Suponiendo que $n-1$ está formado por K bits, el programa retorna en m el numero representado por los primeros K bits aleatorios; es decir: $0 \leq m < 2^K$. Pero la cantidad de bits de $n-1$ es $log2.(n-1)$, por lo tanto la variable entera m retorna un número con distribución uniforme en el intervalo: $\left[0, 2^{log2.n}\right)$. Notar que el programa itera exactamente $log2.(n-1)$ veces, con lo cual no hace uso de alguna terminación probabilística.

Programa $WitnessTail$ **:**

$a' := a^s \texttt{ mod } n;$
$y := 0;$
$witness := false;$
`while` $(y < r \wedge \neg witness)$ `do`
    `if` $(a' \neq 1 \wedge a' \neq n-1)$ `then`
        $a' := a'^2 \texttt{ mod } n;$
        $witness := (a' = 1)$
    `else` $a' := a'^2 \texttt{ mod } n$
    `fi`
    $y := y + 1;$
`od` ;
$witness := witness \vee (a' \neq 1)$

Este fragmento del $MillerRabin$ es necesaria para calcular $a^{n-1} \texttt{ mod } n$ y realizar algunos tests de primalidad en el camino. Notar que el programa calcula la siguiente secuencia:

$$\left(a^{2^0 s} \texttt{ mod } n, a^{2^1 s} \texttt{ mod } n, \ldots, a^{2^r s} \texttt{ mod } n\right)$$

Donde r, s son los naturales encontrados por $FactorTwos$ y que satisfacen $2^r s = n-1$. Si n fuera primo, entonces los numeros de la secuencia deben satisfacer dos test de primalidad:

Test 1 :

$a^{2^r s} \texttt{ mod } n$
$= \{\ 2^r s = n-1\ \}$
$a^{n-1} \texttt{ mod } n$
$= \{\ n$ es primo ; $\phi(n) = n-1\ \}$
$a^{\phi(n)} \texttt{ mod } n$
$= \{\ n$ es primo ; teorema pequeño de Fermat $\}$
1

Test 2 : Para cada $0 < j \leq r$ se tiene que:

$a^{2^j s} \texttt{ mod } n = 1$
$\Rightarrow \{$ algebra $\}$
$0 = (a^{2^j s} - 1) \texttt{ mod } n$
$= ((a^{2^{j-1} s})^2 - 1) \texttt{ mod } n$
$= (a^{2^{j-1} s} + 1)(a^{2^{j-1} s} - 1) \texttt{ mod } n$
$= (a^{2^{j-1} s} + 1) \texttt{ mod } n \ * \ (a^{2^{j-1} s} - 1) \texttt{ mod } n$

$\Rightarrow \{\ n$ es primo $\}$

$(a^{2^{j-1}s}+1)$ mod $n=0 \ \vee \ (a^{2^{j-1}s}-1)$ mod $n=0$
$\Rightarrow$ { algebra }
$a^{2^{j-1}s}$ mod $n=1 \ \vee \ a^{2^{j-1}s}$ mod $n=n-1$

Programa $MillerRabin1$ **:**

$$Uniform_a[2,n-1];$$
$$WitnessTail$$

Este programa genera una base aleatoria a en el rango $[2,n-1]$; luego chequea si es una base atestiguada o no. Es básicamente una iteración del algoritmo $MillerRabin$ principal.

Programa $MillerRabin$ **:**
Luego que se hayan calculado r,s tales que: $2^r s = n-1$ (con $FactorTwos$), se ejecutan T iteraciones del fragmento $MillerRabin1$. En cada una de dichas iteraciones se intenta encontrar alguna base testigo, si esta aparece significa que n no es primo; pero si ello no ocurre solo se puede decir que n es un numero probablemente primo.

2.2 Definiciones

A continuación se definen una función matemática, y algunos predicados que resultarán útiles en la verificación.

1. La función matemática $log2$ se define como:

$$log2 : Int \to Int$$
$$log2.n \ = \ \texttt{if } n=0 \texttt{ then } 0 \texttt{ else } 1+log2.(n \texttt{ div } 2)$$

   Básicamente retorna la cantidad de bits que tiene n .

2. odd.n := n mod $2=1$

3. prime.n := $n>1 \wedge (\forall i : i \geq 1 : \ i|n \ \Rightarrow \ i=1 \vee i=n)$

4. Sean r,s,n naturales tales que: $n-1=2^r s \ \wedge$ odd.s entonces:
   witness.$n.a$:= a^{n-1} mod $n \neq 1 \ \vee$
   $\big(\exists i : 0 \leq i < r : a^{2^{i+1}s}$ mod $n=1 \wedge a^{2^i s}$ mod $n \neq 1 \wedge a^{2^{i+1}s}$ mod $n \neq n-1 \big)$

2.3 Propiedades

Aqui se enuncian las dos propiedades matemáticas fundamentales que se utilizarán en la verificación del algoritmo de $MillerRabin$.

1. $\big(\forall \ n,a : 0<a<n \wedge$ prime.n : $\neg$witness.n.a$\big)$

2. $\left(\forall n : n \geq 3 \wedge \mathsf{odd.n} \wedge \neg \mathsf{prime.n} \ : \ | \{a \ : \ 0 < a < n \wedge \mathsf{witness.n.a} \ \} | \geq \frac{n-1}{2} \right)$

La primer propiedad se deduce inmediatamente de las observaciones hechas en la presentación de $WitnessTail$ cuando n es primo. Notar que para este caso la verificación es sencilla, porque en cada una de las T iteraciones de $MillerRabin$, a nunca será un testigo; con lo cual la varible booleana *prime* finalizará con valor verdadero.

Para el caso de la segunda propiedad, la demostración es un poco más técnica y está detallada en [13]. Notar que la base 1 nunca será un testigo para cada n, luego la busqueda para encontrar algún testigo se restringe al conjunto: $\{2, \ldots, n-1\}$ y sabemos que al menos $\frac{n-1}{2}$ bases de ese conjunto son testigos. Por otra parte tenemos un fragmento de programa que asegura una distribución uniforme en el rango $[2, n-1]$, con lo cual podemos garantizar que la probabilidad de seleccionar un elemento de ese conjunto es de al menos $\frac{1}{n-2}$. Por lo tanto la probabilidad de seleccionar una base testigo será: $(\frac{n-1}{2})(\frac{1}{n-2}) > \frac{1}{2}$.

Capítulo 3

Lógica de Hartog

3.1 Extensión de la lógica de *Hoare*

Sintáxis y semántica de $\mathcal{L}_{pw}$

En primer lugar se presentará el lenguaje $\mathcal{L}_{pw}$ de [4], que abarca los comandos convecionales: asignación, skip, composición secuencial, selección condicional e iteración. Por otra parte se agrega un nuevo comando: *selección probabilística*. Los programas son interpretados como transformadores de estádos, donde el estádo se entiende como la función que mapea variables en valores.

Definición 1 *El conjunto de variables de programa está denotado por PVar y rangeado por x, y. El conjunto de enteros, denotado por Int es extendido con el elemento $\perp$ al conjunto $\text{Int}_\perp$, donde $\perp$ es menor que todos los enteros. El conjunto de estados determinísticos $\mathcal{S}$ rangeado por σ está dado por: $\mathcal{S} = PVar \to Int_\perp$.*

Las operaciones sobre los enteros darán $\perp$ cuando no estén definidas, por ejemplo: división por 0 o cuando alguno de los argumentos no este bien definido.

Ejemplo 1 *Las variables $n, r, s \in PVar$ y σ es un estado determinístico:*
$\sigma = \langle n = 13, r = 2, s = 3 \rangle$. *De esta manera:* $\sigma(n) = 13$.

Definición 2 *El conjunto de expresiones enteras sobre Var, denotado por $Exp\langle \text{Var} \rangle$ es rangueado por e; donde $n \in \text{Int}_\perp$ y Var denota un conjunto de variables, que es rangueado por v.*

$$e \ ::= \ v \mid n \mid e + e \mid e - e \mid e \cdot e \mid e \ \texttt{div} \ e \mid e \ \texttt{mod} \ e \mid e^e$$

La función de valuación que computa el valor de una expresión dados los valores de las variables está definida como:

$$\begin{array}{rcl} \mathcal{V} : Exp\langle Var\rangle \to (Var \to Int_\perp) & \to & Int_\perp \\ \mathcal{V}(n)(f) & = & n \\ \mathcal{V}(v)(f) & = & f(v) \\ \mathcal{V}(e \ \textbf{\textit{op}} \ e')(f) & = & \mathcal{V}(e)(f) \ \textbf{\textit{op}} \ \mathcal{V}(e')(f) \\ \mathcal{V}(e^{e'})(f) & = & \mathcal{V}(e)(f)^{\mathcal{V}(e')(f)} \end{array}$$

$$\textit{donde:} \quad f \in Var \to Int_\perp \quad y \quad \textbf{\textit{op}} \in \{+, -, \cdot, \texttt{mod}\ , \texttt{div}\ \}$$

Notar que en el caso particular que Var=$PVar$, la función f que suple los valores de las variables es un estado $\sigma \in \mathcal{S}$.

Ejemplo 2 *Supongamos que f es una función que povee el valor de las variables r, s; en particular el estado σ definido anteriormente. Entonces el valor de la expresión $2^r s$ es:*
$\mathcal{V}(2^r s) = \mathcal{V}(2^r)(f) \cdot \mathcal{V}(s)(f) = \mathcal{V}(2)(f)^{\mathcal{V}(r)(f)} \cdot f(s) = 2^{f(r)} \cdot f(s) = 2^2 \cdot 3 = 12$

Definición 3 *El conjunto de todas las condiciones booleanas sobre Var es denotado por $BC\langle Var\rangle$ y rangueado por c; donde e es una expresión en $Exp\langle Var\rangle$.*

$$c \ ::= \ true \mid false \mid e = e \mid e < e \mid e \leq e \mid e > e \mid e \geq e \mid c \wedge c \mid c \vee c \mid \neg c \mid c \to c$$

La función de evaluación que computa el valor de una condición booleana dados los valores de las variables provistas por f, está definida por:

$$\begin{array}{rcl} \mathcal{B} : BC\langle Var\rangle \to (Var \to Int_\perp) & \to & Bool \\ \mathcal{B}(true)(f) & = & true \\ \mathcal{B}(false)(f) & = & false \\ \mathcal{B}(\neg c)(f) & = & \neg \mathcal{B}(c)(f) \\ \mathcal{B}(e \ \textbf{\textit{rel}} \ e')(f) & = & \mathcal{V}(e)(f) \ \textbf{\textit{rel}} \ \mathcal{V}(e')(f) \\ \mathcal{B}(c \ \textbf{\textit{op}} \ c')(f) & = & \mathcal{B}(c)(f) \ \textbf{\textit{op}} \ \mathcal{B}(c')(f) \end{array}$$

$$\textit{donde:} \quad f \in Var \to Int_\perp \quad ; \quad \textbf{\textit{rel}} \in \{=, <, \leq, >, \geq\} \quad y \quad \textbf{\textit{op}} \in \{\wedge, \vee, \to\}$$

Ejemplo 3 *Sea $i \in Var$ y $N \in Int$, la condición booleana $(i > 0)$ dice que la variable i es positiva; $(N$ $\texttt{mod}$ $i = 0)$ expresa que N puede ser dividido por i; $(N$ $\texttt{mod}$ $2 = 1)$ indica que N es impar.*

Luego de haber definido las expresiones y condiciones booleanas, ahora se puede definir el lenguaje $\mathcal{L}_{pw}$ de una forma más precisa. Seguidamente se da la gramática que genera los programas del lenguaje.

Definición 4 *Los programas en $\mathcal{L}_{pw}$, rangueados por s, están dados por:*

$$s \ ::= \ \texttt{skip} \mid x := e \mid s; s \mid s \oplus_\rho s \mid \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else}\ s\ \texttt{fi} \mid \texttt{while}\ c\ \texttt{do}\ s\ \texttt{od}$$

donde $x \in PVar$, $e \in Exp\langle PVar\rangle$, $c \in BC\langle PVar\rangle$ y $\rho \in (0, 1)$.

Ejemplo 4 *Un programa sencillo de* $\mathcal{L}_{pw}$ *podría ser:* $b := 0 \oplus_{\frac{1}{2}} b := 1$
Básicamente es un bit aleatorio, cada una de las asignaciones se ejecuta con probabilidad de $\frac{1}{2}$.

Un programa determinístico es interpretado como un tranformador de estados, es decir que el estado resultante de la ejecución del programa está dado por el valor de las variables. Este es el motivo por el cual el espacio de estados $\mathcal{S}$ para programas determinísticos fue definido como: $\mathcal{S} = PVar \rightarrow Int_{\perp}$. En el ejemplo anterior vimos que en un programa probabilítico, los valores de las variables no son más determinados. En lugar de dar el valor de una variable es necesario introducir una distribución sobre todos los posibles valores de esa variable. Además de ello también es necesario considerar la dependecia entre variables de programas; ya que el valor de alguna de ellas podría estar sujeto al valor de otra variable. Estas observaciones motivan a la sigiente definición de estado probabilístico.

Definición 5 *El conjunto de estados probabilísticos* Θ*, rangueado por* θ*, está dado por:*

$$\Theta = \{\ \theta \in \mathcal{S} \rightarrow_{CS} [0,1] \mid \sum_{\sigma \in \mathcal{S}} \theta(\sigma)\ \leq 1\ \}$$

Ejemplo 5 *Supongamos que:* $\sigma_1 = \langle b = 0 \rangle$ *y* $\sigma_2 = \langle b = 1 \rangle$ *entonces un estado probabilśtico podría ser:* $\theta = \frac{1}{3}\sigma_1 + \frac{2}{3}\sigma_2$ *de esta manera:* $\theta(\sigma_1) = \frac{1}{3}$ *y* $\theta(\sigma_2) = \frac{2}{3}$

Aquí $\mathcal{S} \rightarrow_{CS} [0,1]$ denota el conjunto de todas las funciones de $\mathcal{S}$ en [0,1] con soporte contable; es decir que el conjunto $\{\ \sigma \in \mathcal{S} \mid \theta(\sigma) \neq 0\ \}$ es contable. Notar que Θ es un conjunto parcialmente ordenado, donde $\theta \leq \theta'$ si y solo si $\forall \sigma \in \mathcal{S} : \theta(\sigma) \leq \theta'(\sigma)$. El elemento minimal de Θ es θ_0, el estádo probabilístico que le asigna 0 a cada estádo determinístico $\sigma \in \mathcal{S}$. Para una secuencia ascendente en Θ, el limite existe dentro de Θ y corresponde al supremo de la secuencia.

A fin de definir la semántica denotacional del lenguaje $\mathcal{L}_{pw}$, necesitamos previamente dar algunas definiciones. La noción de variante para programas determinísticos se reemplaza por la siguiente extensión probabilística.

Definición 6 *Sean* $f : \mathcal{S} \rightarrow Int_{\perp}$ *y* $\theta \in \Theta$*; el* variante *de* θ*, denotado por* $\theta[x/f]$*, está dado por:*

$$\theta[x/f](\sigma) = \sum_{\sigma' \in V} \theta(\sigma') \quad \textit{donde} \quad V = \{\ \sigma' \in \mathcal{S} \mid \sigma'[x/f(\sigma')] = \sigma\ \}$$

Ejemplo 6 *Como se tiene que:*

$$\begin{aligned}
\langle k = 0 \rangle[k/\mathcal{V}(k \texttt{ div } 2)(\langle k = 0 \rangle)] &= \langle k = 0 \rangle[k/0] = \langle k = 0 \rangle \\
\langle k = 1 \rangle[k/\mathcal{V}(k \texttt{ div } 2)(\langle k = 1 \rangle)] &= \langle k = 1 \rangle[k/0] = \langle k = 0 \rangle \\
\langle k = 2 \rangle[k/\mathcal{V}(k \texttt{ div } 2)(\langle k = 2 \rangle)] &= \langle k = 2 \rangle[k/1] = \langle k = 1 \rangle
\end{aligned}$$

asignando k `div` 2 *a* k *en el estado probabilístico* $\theta = \frac{1}{2}\langle k = 0 \rangle + \frac{1}{4}\langle k = 1 \rangle + \frac{1}{4}\langle k = 2 \rangle$ *da el estado probabilístico:* $\theta[k/\mathcal{V}(k \texttt{ mod } 2)] = \frac{1}{2}\langle k = 0 \rangle + \frac{1}{4}\langle k = 0 \rangle + \frac{1}{4}\langle k = 1 \rangle = \frac{3}{4}\langle k = 0 \rangle + \frac{1}{4}\langle k = 1 \rangle$.

Definición 7

Los operadores de elección probabilística $\oplus_\rho$ y condicional no escalado ? se definen como:

$$\begin{array}{rcll} - \oplus_\rho - & : & \Theta \times \Theta \to \Theta & \rho \in (0,1) \\ \theta_1 \oplus_\rho \theta_2 & = & \rho\theta_1 + (1-\rho)\theta_2 & \end{array}$$

$$\begin{array}{rcll} c? & : & \Theta \to \Theta & c \in BC\langle PVar \rangle \\ c?\theta(\sigma) & = & \left\{ \begin{array}{ll} \theta(\sigma) & si \;\; \mathcal{B}(c)(\sigma) \\ 0 & si \; \neg\mathcal{B}(c)(\sigma) \end{array} \right. & \end{array}$$

El operador $\oplus_\rho$ combina dos estádos probabilísticos con probabilidades apropiadas. El estádo probabilístico $c?\theta$ se obtiene de θ removiendo alguna probabilidad para los estádos determinísticos que no satisfacen la condición c. La suma entre estados probabilísticos se define punto a punto, siempre cuando la probabilidad total de ambos estados no exceda 1.

$$(\theta_1 + \theta_2)(\sigma) = \left\{ \begin{array}{lll} \theta_1(\sigma) + \theta_2(\sigma) & \text{si} & \sum_{\sigma\in\mathcal{S}} \theta_1(\sigma) + \sum_{\sigma\in\mathcal{S}} \theta_1(\sigma) \leq 1 \\ 0 & \text{si} & \sum_{\sigma\in\mathcal{S}} \theta_1(\sigma) + \sum_{\sigma\in\mathcal{S}} \theta_1(\sigma) > 1 \end{array} \right.$$

Notar que cuando la suma total de ambos estados excede 1, se tiene que $\theta_1 + \theta_2 = \theta_0$. Con esta definición de la suma probabilística, las siguientes ecuaciones se satisfacen trivialmente:

$$\begin{array}{rcl} (\theta \oplus_\rho \theta)(\sigma) & = & (\rho\theta + (1-\rho)\theta)(\sigma) \quad = \quad (\rho\theta)(\sigma) + ((1-\rho)\theta)(\sigma) \\ & = & \rho\theta(\sigma) + (1-\rho)\theta(\sigma) \quad = \quad (\rho + 1 - \rho)\theta(\sigma) \quad = \quad \theta(\sigma) \end{array}$$

$$(c?\theta + \neg c?\theta)(\sigma) \;=\; (c?\theta)(\sigma) + (\neg c?\theta)(\sigma) \;=\; \left\{ \begin{array}{ll} \theta(\sigma) + 0 & \text{si} \;\; \mathcal{B}(c)(\sigma) \\ 0 + \theta(\sigma) & \text{si} \; \neg\mathcal{B}(c)(\sigma) \end{array} \right\} \;=\; \theta(\sigma)$$

Por lo tanto: $\theta \oplus_\rho \theta = \theta$ y $c?\theta + \neg c?\theta = \theta$.

Ejemplo 7 *Es simple ver cual es el efecto de los operadores*

$$\begin{array}{rcl} (k \neq 0) \;?\; (\frac{1}{2}\langle k = 0\rangle + \frac{1}{2}\langle k = 1\rangle) & = & \frac{1}{2}\langle k = 1\rangle \\ (k \neq 0) \;?\; 1\langle k = 0\rangle & = & \theta_0 \\ (k \neq 0) \;?\; 1\langle k = 1\rangle & = & 1\langle k = 1\rangle \\ 1\langle b = 2\rangle \oplus_{\frac{1}{2}} (\frac{1}{2}\langle b = 0\rangle + \frac{1}{2}\langle b = 1\rangle) & = & \frac{1}{2}\langle b = 2\rangle + \frac{1}{4}\langle b = 0\rangle + \frac{1}{4}\langle b = 1\rangle \end{array}$$

Bajo estas definiciones, la semántica denotacional de $\mathcal{L}_{pw}$ se define de la siguiente manera:

Definición 8 *La semántica denotacional de $\mathcal{L}_{pw}$ está dada por:*

$$\begin{array}{rcl} \mathcal{D} & : & \mathcal{L}_{pw} \to (\Theta \to \Theta) \\ \mathcal{D}(\texttt{ skip })(\theta) & = & \theta \\ \mathcal{D}(\; x := e \;)(\theta) & = & \theta[x/\mathcal{V}(e)] \\ \mathcal{D}(\; s; s' \;)(\theta) & = & \mathcal{D}(s')\big(\mathcal{D}(s)(\theta)\big) \\ \mathcal{D}(\; s \oplus_\rho s' \;)(\theta) & = & \mathcal{D}(s)(\theta) \oplus_\rho \mathcal{D}(s')(\theta) \\ \mathcal{D}(\texttt{ if } c \texttt{ then } s \texttt{ else } s' \texttt{ fi })(\theta) & = & \mathcal{D}(s)(c?\theta) + \mathcal{D}(s')(\neg c?\theta) \\ \mathcal{D}(\texttt{ while } c \texttt{ do } s \texttt{ od })(\theta) & = & lim_{N\to\infty}\; \neg c \;?\; \mathcal{D}\big((\texttt{ if } c \texttt{ then } s \texttt{ else skip fi })^N\big)(\theta) \end{array}$$

donde: $s^0 := \texttt{skip}$ y $s^{N+1} := s; s^N$

Ejemplo 8 *El siguiente ejemplo es una iteración del ciclo de un programa de 6.6.1 [4]. Supongamos que θ es el siguiente estado probabilístico:*

$$\theta \;=\; \tfrac{1}{2}^{i}\langle x = i, done = false\rangle \;+\; \textstyle\sum_{j=1}^{i} \tfrac{1}{2}^{j}\langle x = j, done = true\rangle$$

aplicando la semántica del programa se tiene:

$$\begin{aligned}
&\mathcal{D}(\texttt{ if } \neg done \texttt{ then } x := x+1;\ done := true \oplus_{\frac{1}{2}} \texttt{skip else skip fi })(\theta)\\
&= \mathcal{D}(\ x := x+1;\ done := true \oplus_{\frac{1}{2}} \texttt{skip })(\neg done?\theta) \;+\; (done?\theta)\\
&= \mathcal{D}(x := x+1)\mathcal{D}(\ done := true \oplus_{\frac{1}{2}} \texttt{skip }\,)(\neg done?\theta) \;+\; (done?\theta)\\
&= \mathcal{D}(x := x+1)\big(\ \mathcal{D}(done := true)(\neg done?\theta) \;\oplus_{\frac{1}{2}}\; \mathcal{D}(\texttt{ skip })(\neg done?\theta)\ \big) \;+\; (done?\theta)\\
&= \mathcal{D}(x := x+1)\big(\ (\neg done?\theta)[done/\mathcal{V}(true)] \;\oplus_{\frac{1}{2}}\; (\neg done?\theta)\ \big) \;+\; (done?\theta)\\
&= \mathcal{D}(x := x+1)\big(\ \tfrac{1}{2}^{i}\langle x = i, done = false\rangle[done/\mathcal{V}(true)] \;\oplus_{\frac{1}{2}}\; \tfrac{1}{2}^{i}\langle x = i, done = false\rangle\ \big) \;+\; (done?\theta)\\
&= \mathcal{D}(x := x+1)\big(\ \tfrac{1}{2}^{i+1}\langle x = i, done = true\rangle \;+\; \tfrac{1}{2}^{i+1}\langle x = i, done = false\rangle\ \big) \;+\; (done?\theta)\\
&= \big(\ \tfrac{1}{2}^{i+1}\langle x = i, done = true\rangle \;+\; \tfrac{1}{2}^{i+1}\langle x = i, done = false\rangle\ \big)[x/\mathcal{V}(x+1)] \;+\; (done?\theta)\\
&= \tfrac{1}{2}^{i+1}\langle x = i{+}1, done = true\rangle \;+\; \tfrac{1}{2}^{i+1}\langle x = i{+}1, done = false\rangle \;+\; \textstyle\sum_{j=1}^{i} \tfrac{1}{2}^{j}\langle x = j, done = true\rangle\\
&= \tfrac{1}{2}^{i+1}\langle x = i{+}1, done = false\rangle \;+\; \textstyle\sum_{j=1}^{i+1} \tfrac{1}{2}^{j}\langle x = j, done = true\rangle
\end{aligned}$$

Ejemplo 9 *Este ejemplo tiene como objetivo clarificar un poco la semántica del* `while` *, que quizás sea la menos intuitiva. Se le dará semántica al programa* `while` $(k \neq 0)$ `do` $k := k$ `div 2 od` *partiendo del estado* $\langle k = 2^M\rangle$ *. En primer lugar se probará por inducción en N dos propiedades auxliares; donde N es el numero de iteración.*

- *(1)* $\forall\ N \geq 1 : \mathcal{D}(\texttt{if }^{N})\langle k = 2^N\rangle \;=\; \langle k = 1\rangle$

  caso $N = 1$

  $\mathcal{D}(\texttt{ if }^{1})\langle k = 2\rangle$

  $= \mathcal{D}(\texttt{ if } (k \neq 0) \texttt{ then } k := k \texttt{ div } 2 \texttt{ else skip fi })\langle k = 2\rangle$

  $= \mathcal{D}(k := k \texttt{ div } 2)((k \neq 0)?\langle k = 2\rangle) \;+\; \mathcal{D}(\texttt{skip })(\neg(k \neq 0)?\langle k = 2\rangle)$

  $= \mathcal{D}(k := k \texttt{ div } 2)\langle k = 2\rangle \;+\; \mathcal{D}(\texttt{skip })(\theta_0)$

  $= \langle k = 2\rangle[k/\mathcal{V}(k \texttt{ div } 2)] \;+\; \theta_0$

  $= \langle k = 1\rangle \quad = \mathcal{D}(k := k \texttt{ div } 2)\langle k = 2\rangle \;+\; \mathcal{D}(\texttt{skip })(\theta_0)$

  $= \langle k = 2\rangle[k/\mathcal{V}(k \texttt{ div } 2)] \;+\; \theta_0$

  $= \langle k = 1\rangle \quad \diamond$

  caso $N + 1$

  $\mathcal{D}(\texttt{ if }^{N+1})\langle k = 2^{N+1}\rangle$

  $= \mathcal{D}(\texttt{ if }^{N}\texttt{; if }\,)\langle k = 2^{N+1}\rangle$

  $= \mathcal{D}(\texttt{ if }^{N})\mathcal{D}(\texttt{ if }\,)\langle k = 2^{N+1}\rangle$

  $= \mathcal{D}(\texttt{ if }^{N})\big(\ \mathcal{D}(k := k \texttt{ div } 2)((k \neq 0)?\langle k = 2^{N+1}\rangle) \;+\; \mathcal{D}(\texttt{skip })(\neg(k \neq 0)?\langle k = 2^{N+1}\rangle)\ \big)$

  $= \mathcal{D}(\texttt{ if }^{N})\big(\ \mathcal{D}(k := k \texttt{ div } 2)\langle k = 2^{N+1}\rangle \;+\; \mathcal{D}(\texttt{skip })(\theta_0)\ \big)$

  $= \mathcal{D}(\texttt{ if }^{N})\big(\ \langle k = 2^{N+1}\rangle[k/\mathcal{V}(k \texttt{ div } 2)] \;+\; \theta_0\ \big)$

  $= \mathcal{D}(\texttt{ if }^{N})\langle k = 2^{N}$

  $= \langle k = 1\rangle \qquad \longleftarrow$ *hipotesis inductiva* $\diamond$

- *(2)* $\forall\ N \geq M+1 : \mathcal{D}(\texttt{if}^{\ N})\langle k = 2^M\rangle \ = \ \langle k = 0\rangle$

  caso $N = M+1$

  $\mathcal{D}(\ \texttt{if}^{\ M+1})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ ;\texttt{if}^{\ M})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ )\mathcal{D}(\ \texttt{if}^{\ M})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ )\langle k = 1\rangle \qquad \longleftarrow$ *por (1)*
  $= \mathcal{D}(k := k\ \texttt{div}\ 2)((k \neq 0)?\langle k = 1\rangle) \ + \ \mathcal{D}(\texttt{skip}\ )(\neg(k \neq 0)?\langle k = 1\rangle)$
  $= \mathcal{D}(k := k\ \texttt{div}\ 2)\langle k = 1\rangle \ + \ \mathcal{D}(\texttt{skip}\ )(\theta_0)$
  $= \langle k = 1\rangle[k/\mathcal{V}(k\ \texttt{div}\ 2)] \ + \ \theta_0$
  $= \langle k = 0\rangle \quad \diamond$

  caso $N+1$

  $\mathcal{D}(\ \texttt{if}^{\ N+1})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ ;\texttt{if}^{\ N})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ )\mathcal{D}(\texttt{if}^{\ N})\langle k = 2^M\rangle$
  $= \mathcal{D}(\ \texttt{if}\ )\langle k = 0\rangle \qquad \longleftarrow$ *hipotesis inductiva*
  $= \mathcal{D}(k := k\ \texttt{div}\ 2)((k \neq 0)?\langle k = 0\rangle) \ + \ \mathcal{D}(\texttt{skip}\ )(\neg(k \neq 0)?\langle k = 0\rangle)$
  $= \mathcal{D}(k := k\ \texttt{div}\ 2)(\theta_0) \ + \ \mathcal{D}(\texttt{skip}\ )\langle k = 0\rangle$
  $= \theta_0 + \langle k = 0\rangle$
  $= \langle k = 0\rangle \quad \diamond$

- $\mathcal{D}(\ \texttt{while}\ (k \neq 0)\ \texttt{do}\ k := k\ \texttt{div}\ 2\ \texttt{od}\ )\langle k = 2^M\rangle$
  $= \lim_{N\to\infty} \neg(k \neq 0)\ ?\ \mathcal{D}(\texttt{if}^{\ N})\langle k = 2^M\rangle$
  $= \neg(k \neq 0)\ ?\ \langle k = 0\rangle \qquad \longleftarrow$ *por (2)*
  $= \langle k = 0\rangle \quad \diamond$

Puede probarse que la semantica es lineal con respecto al estádo probabilístico (ver lema 12 del apéndice), es decir que vale: $\mathcal{D}(s)(\rho \cdot \theta + \theta') \ = \ \rho\mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta')$. Sin embargo en general $\mathcal{D}(s)(c?\theta) \ \neq \ c?\mathcal{D}(s)(\theta)$. Por ejemplo:

$$\mathcal{D}(x := 1)((x = 0)?\langle x = 0\rangle) \ = \ \mathcal{D}(x := 1)\langle x = 0\rangle \ = \ \langle x = 0\rangle[x/\mathcal{V}(1)] \ = \ \langle x = 1\rangle$$
$$\langle x = 0\rangle\ ?\ \mathcal{D}(x := 1)\langle x = 0\rangle \ = \ \langle x = 0\rangle\ ?\ \langle x = 0\rangle[x/\mathcal{V}(1)] \ = \ \langle x = 0\rangle\ ?\ \langle x = 1\rangle \ = \ \theta_0$$

Predicados Probabilísticos y extensión de Lógica de Hoare

En está sección se dará la lógica probabilística presentada en [4], en la cual los predicados especifican propiedades de estados probabilísticos.

Para deducir cuales triplas $\{p\}s\{q\}$ de *Hoare* son válidas, se introduce un sistema de prueba *pH*. Este sistema de prueba se basa en la lógica estándard de *Hoare* para programas no probabilísticos. Las reglas conocidas de la lógica tradicional son adaptadas introduciendo probabilidad, y varias reglas nuevas son añadidas. Está demostrado que este sistema de prueba es correcto y completo.

Definición 9 *Sea IVar un conjunto de variables valuadas enteras, rangueadas por i. El conjunto de* predicados determinísticos *es denotado DPred y rangueado por dp.*

$$\begin{aligned} dp \ ::= \ & true \mid false \mid e = e \mid e < e \mid e \leq e \mid e > e \mid e \geq e \mid \\ & dp \wedge dp \mid dp \vee dp \mid \neg dp \mid dp \rightarrow dp \mid \forall i : dp \mid \exists i : dp \end{aligned}$$

Notar que $e \in Exp\langle PVar \cup IVar\rangle$ *es una expresión sobre variables de programa y variables valuadas enteras. El conjunto* $\mathcal{I}$ *de todas las interpretaciones de variables valuadas enteras está dado por* $\mathcal{I} = IVar \rightarrow Int_\perp$. *Una interpretación típica se denota con I. La relación de satisfacción para predicados determinísticos, se define de la siguiente manera:*

$$\begin{array}{lcl} \_ \vDash \_ \ : \ (\mathcal{S} \times \mathcal{I}) \times DPred \quad \rightarrow \quad Bool & & \\ (\sigma, I) \vDash true & & \\ (\sigma, I) \nvDash false & & \\ (\sigma, I) \vDash \neg dp & \iff & \neg(\sigma, I) \vDash dp \\ (\sigma, I) \vDash e \ \textbf{\textit{rel}} \ e' & \iff & \mathcal{V}(e)(\sigma, I) \ \textbf{\textit{rel}} \ \mathcal{V}(e')(\sigma, I) \\ (\sigma, I) \vDash dp \ \textbf{\textit{op}} \ dp' & \iff & (\sigma, I) \vDash dp \ \textbf{\textit{op}} \ (\sigma, I) \vDash dp' \\ (\sigma, I) \vDash (\forall i : dp) & \iff & \forall n \in Int_\perp : (\sigma, I[i/n]) \vDash dp \\ (\sigma, I) \vDash (\exists i : dp) & \iff & \exists n \in Int_\perp : (\sigma, I[i/n]) \vDash dp \end{array}$$

donde: $\textbf{\textit{rel}} \in \{=, <, \leq, >, \geq\}$ *y* $\textbf{\textit{op}} \in \{\wedge, \vee, \rightarrow\}$

Ejemplo 10 *Como* $(i \geq 1 \wedge x \ \texttt{mod} \ i = 0) \rightarrow (i = 1 \vee i = x)$ *se satisface en* $(\langle x = 3\rangle, \langle i = n\rangle)$ *para todo* $n \in Int$, *ya que* 3 *es un numero primo; entonces tenemos que para toda interpretación I:*
$(\sigma, I) \vDash \ \forall i : (i \geq 1 \wedge x \ \texttt{mod} \ i = 0) \rightarrow (i = 1 \vee i = x)$

Notar que tanto el estádo determinístico σ, como la interpretación I de variables valuadas enteras son necesarios para evaluar un predicado determinístico.

Definición 10 *1. Una interpretación J de variables valuadas enteras y reales es una función que asigna un elemento de* $\text{Int}_\perp$ *a cada variable valuada entera y un elemento de* $\mathbb{R}_\perp$ *a cada variable valuada real. El conjunto* $\mathbb{R}_\perp$ *es el los reales extendido de manera usual con el elemento* $\perp$ *denotando indefinición. La restricción de J a las variables valuadas enteras está denotada por* $J_\mathcal{I}$; *y el conjunto de todas las interpretaciones se denota con* $\mathcal{J}$.

2. *Sea RVar el conjunto de variables valuadas reales, ranguedas por r. El conjunto de expresiones reales RealExp está rangueado por e_r.*

$$e_r \ ::= \ \rho \mid r \mid \mathbb{P}(dp) \mid e_r + e_r \mid e_r - e_r \mid e_r * e_r \mid e_r / e_r \mid e_r^e$$

Donde $\rho \in \mathbb{R}$ *y* $e \in \langle IVar \rangle$ *. La función de evaluación para expresiones reales está dada por:*

$$\begin{array}{rcl} \mathcal{V}_r : RealExp \to (\Theta \times \mathcal{J}) & \to & \mathbb{R}_\perp \\ \mathcal{V}_r(\rho)(\theta, J) & = & \rho \\ \mathcal{V}_r(r)(\theta, J) & = & J(r) \\ \mathcal{V}_r(\mathbb{P}(dp))(\theta, J) & = & \sum_{\sigma \in V} \theta(\sigma) \\ \mathcal{V}_r(e_r \ \textbf{\textit{op}} \ e'_r)(\theta, J) & = & \mathcal{V}_r(e_r)(\theta, J) \ \textbf{\textit{op}} \ \mathcal{V}_r(e'_r)(\theta, J) \\ \mathcal{V}_r(e_r{}^{e'})(\theta, J) & = & \mathcal{V}_r(e_r)(\theta, J)^{\mathcal{V}(e')(J_\mathcal{I})} \end{array}$$

donde: $V = \{\sigma' | (\sigma', J_\mathcal{I}) \vDash dp\}$ *y* $\textbf{\textit{op}} \in \{+, -, *, /\}$

3. *La metavariable j es usada para ranguear sobre la unión del conjunto de variables valuadas enteras y el conjunto de varibles valuadas reales:* $j \ ::= \ i \mid r$.

Ejemplo 11 *En el estado probabilístico* $\theta = \frac{1}{3}\langle m = 0 \rangle + \frac{1}{3}\langle m = 1 \rangle + \frac{1}{3}\langle m = 2 \rangle$ *y con la interpretación* $J = \langle i = 2, r = \frac{1}{4} \rangle$*, la expresión* $r * \mathbb{P}(0 \leq m < i)$ *evalúa a:* $\frac{1}{4}(\frac{1}{3} + \frac{1}{3}) = \frac{1}{6}$.

La distribución de las variables de programa está dada por un estado probabilístico θ, esa es la razón por la cual una variable de programa no puede ser usada fuera del contructor $\mathbb{P}(\cdot)$, ya que el valor de una variable no es fijado en el estado probabilístico, solo la distribución. Notar que el valor de $\mathbb{P}(dp)$ es la probabilidad que el predicado determinístico dp sea válido en el estado probabilístico θ. Esta probabilidad se obtiene sumando las probabilidades de todos los estádos determinísticos que satisfacen dp.

Definición 11 *El conjunto de condiciones reales, denotado por RC está rangueado por* c_r.

$$c_r \ ::= \ c \mid e_r = e_r \mid e_r < e_r \mid e_r \leq e_r \mid e_r > e_r \mid e_r \geq e_r \mid c_r \wedge c_r \mid c_r \vee c_r \mid \neg c_r \mid c_r \to c_r$$

Donde $c \in BC\langle IVar \rangle$*. La función de evaluación que computa el valor de una condición real, dados los valores de las variables, está dado por:*

$$\begin{array}{rcl} \mathcal{B}_r : RC \to (\Theta \times \mathcal{J}) & \to & Bool \\ \mathcal{B}_r(c)(\theta, J) & = & \mathcal{B}(c)(J_\mathcal{I}) \\ \mathcal{B}_r(\neg c)(\theta, J) & = & \neg \mathcal{B}_r(c)(\theta, J) \\ \mathcal{B}_r(e \ \textbf{\textit{rel}} \ e')(\theta, J) & = & \mathcal{V}_r(e)(\theta, J) \ \textbf{\textit{rel}} \ \mathcal{V}_r(e')(\theta, J) \\ \mathcal{B}_r(c \ \textbf{\textit{op}} \ c')(\theta, J) & = & \mathcal{B}_r(c)(\theta, J) \ \textbf{\textit{op}} \ \mathcal{B}_r(c')(\theta, J) \end{array}$$

donde: $\textbf{\textit{rel}} \in \{=, <, \leq, >, \geq\}$ *y* $\textbf{\textit{op}} \in \{\wedge, \vee, \to\}$

Ejemplo 12 *Un ejemplo de condiciones basadas en reales es* $\mathbb{P}(x = 0) = 1 \vee \mathbb{P}(x = 1) = 1$ *. Notar que el estado probabilístico* $1\langle x = 1 \rangle$ *la satisface, pero no así el estado* $\frac{1}{2}\langle x = 0 \rangle + \frac{1}{2}\langle x = 1 \rangle$

Definición 12 *El conjunto de predicados probabilísticos Pred está rangueado por p y q.*

$$p \ ::= \ c_r \mid p \wedge p \mid p \vee p \mid \neg p \mid p \to p \mid \exists j : p \mid \forall j : p \mid \rho \cdot p \mid p + p \mid p \oplus_\rho p \mid c?p$$

Donde $c \in BC\langle PVar\rangle$. La relación de satisfacción para predicados probabilísticos está dada por:

$$\begin{array}{lcl}
_ \vDash _ & : & (\Theta \times \mathcal{J}) \times Pred \to Bool \\
(\theta, J) \vDash c_r & \Longleftrightarrow & \mathcal{B}_r(c_r)(\theta, J) \\
(\theta, J) \vDash \rho \cdot p & \Longleftrightarrow & \exists \theta' : \theta = \rho \cdot \theta' \wedge (\theta', J) \vDash p \\
(\theta, J) \vDash p + p' & \Longleftrightarrow & \exists \theta_1, \theta_2 : \theta = \theta_1 + \theta_2 \wedge (\theta_1, J) \vDash p \wedge (\theta_2, J) \vDash p' \\
(\theta, J) \vDash p \oplus_\rho p' & \Longleftrightarrow & \exists \theta_1, \theta_2 : \theta = \rho\theta_1 + (1-\rho)\theta_2 \wedge (\theta_1, J) \vDash p \wedge (\theta_2, J) \vDash p' \\
(\theta, J) \vDash c?p & \Longleftrightarrow & \exists \theta' : \theta = c?\theta' \wedge (\theta', J) \vDash p \\
(\theta, J) \vDash \neg p & \Longleftrightarrow & \neg(\theta, J) \vDash p \\
(\theta, J) \vDash p \ \boldsymbol{op} \ p' & \Longleftrightarrow & (\theta, J) \vDash p \ \boldsymbol{op} \ (\theta, J) \vDash p' \\
(\theta, J) \vDash \forall j : p & \Longleftrightarrow & \forall n \in Int_\perp : (\theta, J[j/n]) \vDash p \\
(\theta, J) \vDash \exists j : p & \Longleftrightarrow & \exists n \in Int_\perp : (\theta, J[j/n]) \vDash p
\end{array}$$

donde: $\boldsymbol{op} \in \{\wedge, \vee, \to\}$

El operador $+$ es usado para la adición de predicados, el operador $\rho\cdot$ sirve para escalar. El operador $\oplus_\rho$ combina adición y escalado; $c?p$ da probabilidades condicionales sin normalizar. Notar que dos predicados probabilísticos p y q son *equivalentes* si y solo si:

$$\forall \theta \in \Theta, J \in \mathcal{J} \ : \ (\theta, J) \vDash p \iff (\theta, J) \vDash q$$

Ejemplo 13 *El predicado $(\forall i : 0 \leq i < N \to \mathbb{P}(m = i) = \frac{1}{N})$ dice que la variable m está uniformemente distribuida en el intervalo $[0, N)$. En la sección 3.3 se verifica justamente un programa cuya variable m tiene una distribución uniforme. Si definimos $p := \mathbb{P}(x = 0) = 1 \vee \mathbb{P}(x = 1) = 1$ entonces el predicado $p \oplus_{\frac{1}{2}} p$ es satisfecho por el estado probabilístico $\theta = \frac{1}{2}\langle x = 0\rangle + \frac{1}{2}\langle x = 1\rangle$; sin embargo según el ejemplo 12 este mismo estado no satisfacía el predicado p; lo cual verifica que: $(\theta, J) \vDash p \oplus_\rho p \not\Rightarrow (\theta, J) \vDash p$.*

A fin de probar la validez de triplas de *Hoare* del estilo $\{\ p\ \}\ s\ \{\ q\ \}$, se introduce el sistema de prueba *pH*. El sistema de prueba consiste de axiomas y reglas como las siguientes:

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

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

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

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

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

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

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

Notar que las reglas (Skip), (Assign),(Seq) y (Cons) son como las de la lógica de *Hoare* tradicional, pero empleando predicados probabilísticos. Las reglas (If) y (While) han sido cambiadas. Las nuevas reglas introducidas son: (Prob), (Or), (And), (Exists),(Forall), (Lin +), (Lin ·). Notar que $FV(p)$ representa el conjunto de variables libres, es decir, las que no están ligadas a ningún cuantificador $\exists$, $\forall$. En la lógica tradicional de *Hoare*, la regla Más adelante se dará una definición precisa y un ejemplo acerca de la condición $\langle c, s\rangle$-closed para el invariante p en la regla (While).

Ejemplo 14 *El siguiente es un ejemplo de un arbol de prueba en el sistema pH. Usando las reglas en pH, se puede obtener la siguiente tripla de Hoare:*

$$\{\ \mathbb{P}(I) = 1\ \}\ UnifPow2_n\ \{\ \left(\forall j : 0 \leq j < 2^{log2.N} : \mathbb{P}(m = j \wedge I) = 2^{-log2.N}\right)\ \}$$

donde $I := n = N$ *y* $UnifPow2_n$ *es el siguiente programa, cuya variable* m *resulta tener una distribución uniforme en el intervalo:* $\left[0, 2^{log2.N}\right)$.

$$\left.\begin{array}{l} k := n; \\ m := 0; \\ \texttt{while}\ (k \neq 0)\ \texttt{do} \\ \quad k := k\ \texttt{div}\ 2; \\ \quad b := 0 \oplus_{\frac{1}{2}} b := 1; \\ \quad m := 2m + b \\ \texttt{od} \end{array}\right\} UnifPow2_n$$

A fin de hacer má legible la derivación del arbol se dará una prueba outline. Primeramente definimos el invariante probabilístico Inv *del ciclo, y un invariante estandard* I *que no es modificado por las variables del programa* $UnifPow2_n$.

$$\begin{array}{rcl} Inv & := & (\exists i : q) \\ q & := & \left(\forall j : 0 \leq j < 2^{log2.N - log2.i} : \mathbb{P}(m = j \wedge k = i \wedge I) \geq 2^{-log2.N + log2.i}\right) \\ I & := & n = N \end{array}$$

Notar que $2^{-log2.N+log2.k}$ *no sería una expresión real correcta, ya que las variables de programa, en particular la variable* k*, no pueden ser usadas en expresiones reales, excepto dentro de un predicado determinístico en el constructor* $\mathbb{P}(\cdots)$*. Con lo cual es necesario introducir un* $\exists$*, a fin de expresar:*

$$(\exists i : \cdots \mathbb{P}(k = i \wedge \cdots) \geq 2^{-log2.N+log2.i} \cdots)$$

A continuación se prueba la inicialización y salida del ciclo de $UnifPow2_n$*. Para probar que* Inv *es invariante del ciclo, se hace uso de la regla (Exists) aplicada a la terna:*

$$\{\ q\ \}\ \texttt{if}\ (k \neq 0)\ \texttt{then}\ \ k := k\ \texttt{div}\ 2;\ b := 0 \oplus_{\frac{1}{2}} b := 1;\ m := 2m + b\ \texttt{else skip fi}\ \ \{\ Inv\ \}$$

que será demostrada luego.

$\{\ \mathbb{P}(I) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(0 = 0 \wedge n = N \wedge I) = 1\ \}$

$k := n;$

$\{\ \mathbb{P}(0 = 0 \wedge k = N \wedge I) = 1\ \}$

$m := 0;$

$\{\ \mathbb{P}(m = 0 \wedge k = N \wedge I) = 1\ \}$

$\Rightarrow \{\ Inv\ \}$

`while` $(k \neq 0)$ `do`

$k := k$ `div` $2;$

$b := 0 \oplus_{\frac{1}{2}} b := 1;$

$m := 2m + b$

`od`

$\{\ \mathbb{P}(k \neq 0) = 0 \wedge Inv\ \}$

$\Rightarrow \{\ \left(\forall j : 0 \leq j < 2^{log2.N} : \mathbb{P}(m = j \wedge I) \geq 2^{-log2.N}\right)\ \}$

$\Rightarrow \{\ \left(\forall j : 0 \leq j < 2^{log2.N} : \mathbb{P}(m = j \wedge I) = 2^{-log2.N}\right)\ \}$

En la inicialización, se usó dos veces la regla (Assign) y luego se aplicó la regla (Cons). La primer implicación en la salida del ciclo es clara por el hecho que para algún valor $i \neq 0$, el predicado

$$\mathbb{P}(m = j \wedge k = i \wedge I) \geq 2^{-log2.N+log2.i}$$

implica alguna probabilidad positiva para $k \neq 0$. La ultima implicación es consecuencia de que si valiera el $>$ estricto en alguna de las $2^{log2.N}$ desigualdades entonces tendriamos que:

$$\mathbb{P}(I) = \sum_j \mathbb{P}(m = j \wedge I) > 2^{log2.N} 2^{-log2.N} = 1$$

Lo cual es un absurdo, por lo tanto vale el $=$ en las $2^{log2.N}$ desigualdades.

$\{\ q\ \}$

`while` $(k \neq 0)$ `do`

$\{\ (k \neq 0)?q\ \}$

$\Rightarrow \{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2) - 1}\ :$

$\mathbb{P}(m = j \wedge k\ \texttt{div}\ 2 = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2) + 1}\big)\ \}$

$k := k\ \texttt{div}\ 2;$

$\{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2) - 1}\ :$

$\mathbb{P}(m = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2) + 1}\big)\ \}$

$b := 0 \oplus_{\frac{1}{2}} b := 1;$

$\{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)} \wedge \mathsf{even}.j\ \ :$

$\mathbb{P}(2 * m + b = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2) + 1}\big)\ \oplus_{\frac{1}{2}}$

$\big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)} \wedge \mathsf{odd}.j\ \ :$

$\mathbb{P}(2 * m + b = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2) + 1}\big)\ \}$

$\Rightarrow \{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)} \wedge \mathsf{even}.j\ \ :$

$\mathbb{P}(2 * m + b = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2)}\big)\ \wedge$

$\big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)} \wedge \mathsf{odd}.j\ \ :$

$\mathbb{P}(2 * m + b = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2)}\big)\ \}$

$\Rightarrow \{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)}\ :$

$\mathbb{P}(2m + b = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2)}\big)\ \}$

$m := 2m + b$

$\{\ \big(\forall j : 0 \leq j < 2^{log2.N - log2.(i\ \texttt{div}\ 2)}\ :$

$\mathbb{P}(m = j \wedge k = i\ \texttt{div}\ 2 \wedge I) \geq 2^{-log2.N + log2.(i\ \texttt{div}\ 2)}\big)\ \}$

$\Rightarrow \{\ q[i/i\ \texttt{div}\ 2]\ \}$

$\Rightarrow \{\ Inv\ \}$

$\{\neg(k \neq 0)?q\ \}$

`skip`

$\{\ true\ \}$

`od`

$\{\ Inv + true\ \}$

$\Rightarrow \{\ Inv\ \}$

En la primer implicación se utiliza que $i \neq 0$ *imlplica que* $log2.i = log2.(i\ \texttt{div}\ 2) + 1$. *Seguidamente se aplica la regla (Assign) y luego la regla (Prob), donde las dos subpruebas correspondientes a la premisa, son detalladas a continuación:*

•

$\{\ (\forall j : 0 \leq j < 2^{log2.N - log2.(i \texttt{ div } 2) - 1} :$
$\mathbb{P}(m = j \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$\Rightarrow \{\ (\forall j : 0 \leq 2 * j < 2^{log2.N - log2.(i \texttt{ div } 2)} :$
$\mathbb{P}(2 * m + 0 = 2 * j \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$b := 0;$
$\{\ (\forall j : 0 \leq 2 * j < 2^{log2.N - log2.(i \texttt{ div } 2)} :$
$\mathbb{P}(2 * m + b = 2 * j \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$\Rightarrow \{\ (\forall j' : 0 \leq j' < 2^{log2.N - log2.(i \texttt{ div } 2)} \wedge \mathsf{even}.j' :$
$\mathbb{P}(2 * m + b = j' \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$

•

$\{\ (\forall j : 0 \leq j < 2^{log2.N - log2.(i \texttt{ div } 2) - 1} :$
$\mathbb{P}(m = j \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$\Rightarrow \{\ (\forall j : 0 \leq 2 * j < 2^{log2.N - log2.(i \texttt{ div } 2)} :$
$\mathbb{P}(2 * m + 1 = 2 * j + 1 \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$b := 1;$
$\{\ (\forall j : 0 \leq 2 * j < 2^{log2.N - log2.(i \texttt{ div } 2)} :$
$\mathbb{P}(2 * m + b = 2 * j + 1 \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$
$\Rightarrow \{\ (\forall j' : 0 \leq j' < 2^{log2.N - log2.(i \texttt{ div } 2)} \wedge \mathsf{odd}.j' :$
$\mathbb{P}(2 * m + b = j' \wedge k = i \texttt{ div } 2 \wedge I) \geq 2^{-log2.N + log2.(i \texttt{ div } 2) + 1})\ \}$

En la tercer implicación, cada una de las probabilidades $2^{-log2.N + log2.i + 1}$ *que figuran a la izquierda y derecha del operador* $\oplus_{\frac{1}{2}}$ *se multiplica por* $\frac{1}{2}$*. La cuarta implicación es una partición de rango. Finalmente se aplica la regla (Assign). Las ultima implicaciónes de la rama positiva del* `if` *son inmediatas por definición de q e Inv. Sabemos que true se satisface en cualquier parte, en particular en la rama negativa del* `if` *tenemos que vale la tripla:* $\{\neg(k \neq 0)\}$`skip` $\{true\}$*. Notar que cuando combinamos ambos resultados con el operador +, usando el lema 9 del apéndice, el invariante no se altera, ya que en el peor de los casos, la probabilidad se incrementa aun mas, manteniéndose el* $\geq$*.*

fin Ejemplo

A fin de asegurar corrección total, la lógica de *Hartog*, refuerza la noción de invariante imponiendo una condición extra denominada: $\langle c, s\rangle$-closed. Básicamente expresa que una secuencia de estados probabilísticos que satisface el estado p, que podrá ser obtenida por repetidas iteraciones del ciclo `while` , debe tener un límite también satisfaciendo p.

Definición 13 *$\langle c, s\rangle$-closed*

1. *Para un predicado p la razón de terminación del m-ésimo paso, denotada por* $r^m_{\langle c,s\rangle}$*, es la mínima probalidad que, empezando de un estádo satisfaciendo p, el ciclo* `while` c `do` s`od` *termine en m*

pasos.

$$\begin{array}{rcl} r(\theta)^m_{\langle c,s\rangle} & = & \sum_{\sigma\in\mathcal{S}} \left[\neg c \ ?\ \mathcal{D}\big((\texttt{if } c \texttt{ then } s \texttt{ else skip fi })^m\big)(\theta)\ \right]\ (\sigma) \\ r^m_{\langle c,s\rangle} & = & inf\{\ r(\theta)^m_{\langle c,s\rangle}\ \mid (\theta, J) \vDash p\ \} \end{array}$$

2. *Una secuencia de estados $(\theta_m)_{m\in\mathbb{N}}$ se denomina una $\langle c,s\rangle$-secuencia dentro de p si $(\neg c?\theta_m)_{m\in\mathbb{N}}$ es una cadena ascendente, para cada $m \in \mathbb{N}$ el estádo θ_n satisface p y se tiene que*

$$\sum_{\sigma\in\mathcal{S}} (\neg c?\theta_m)(\sigma) \ \geq\ r^m_{\langle c,s\rangle}$$

3. *Una $\langle c,s\rangle$-secuencia $(\theta_m)_{m\in\mathbb{N}}$ en p se dice que termina en p si el supremo de la secuencia $(\neg c?\theta_m)_{m\in\mathbb{N}}$ satisface p.*

4. *Un predicado p es llamado $\langle c,s\rangle$-closed si cada $\langle c,s\rangle$-secuencia en p termina en p.*

La noción de $\langle c,s\rangle$-closedness expresa que una secuencia que puede ser obenida por repetidas iteraciones del ciclo `while` c `do` s `od` tendrá un supremo satisfaciendo p. Notar que para un ciclo que termina, cada predicado p automaticamente satisface $\langle c,s\rangle$-closedness, con lo cual no es necesario chequear esas condiciones. El siguiente ejemplo chequea la condición $\langle c,s\rangle$-closed para el invariante Inv del ejemplo 16.

Ejemplo 15 *En el ejemplo 16 demostramos la invarianza de Inv respecto del ciclo, lo que faltaba chequear es que Inv sea $\langle c,s\rangle$-closed. Asumamos alguna interpretación fija J. Calculemos en primer lugar la razón de terminación en m-pasos. Para un estado que satisface q usamos la tripla de Hoare deducida anteriormente:*

$$\{\ q\ \}\ \texttt{if}\ (k \neq 0)\ \texttt{then}\ \ k := k\ \texttt{div}\ 2;\ \ b := 0 \oplus_{\frac{1}{2}} b := 1;\ \ m := 2m + b\ \texttt{else skip fi}\ \ \{\ q[i/i\ \texttt{div}\ 2]\ \}$$

Esta tripla dice que si: $(\theta, J) \vDash q$ *entonces* $\forall m \in Int :\ (\texttt{if}^{\ m}_{\langle c,s\rangle}(\theta), J) \vDash q[i/i\ \texttt{div}\ 2^m]$.
Como $i \leq N$ *y* $N\ \texttt{div}\ 2 = 0 \ \Leftrightarrow\ m \geq log2.N$ *; luego* $q[i/i\ \texttt{div}\ 2^m]$ *implica que:*

$$\mathbb{P}(k=0) = \left\{ \begin{array}{ll} 0 & si\ i\ \texttt{div}\ 2^m \neq 0 \\ 1 & si\ i\ \texttt{div}\ 2^m = 0 \end{array} \right. \geq \left\{ \begin{array}{ll} 0 & si\ N\ \texttt{div}\ 2^m \neq 0 \\ 1 & si\ N\ \texttt{div}\ 2^m = 0 \end{array} \right. = \left\{ \begin{array}{ll} 0 & si\ m < log2.N \\ 1 & si\ m \geq log2.N \end{array} \right.$$

con lo cual tenemos que

$$(\theta, J) \vDash q \quad \Rightarrow \quad r(\theta)^m_{\langle c,s\rangle} \geq \left\{ \begin{array}{ll} 0 & si\ m < log2.N \\ 1 & si\ m \geq log2.N \end{array} \right.$$

Por lo tanto para todos los estados que satisfacen el predicado Inv y para todo $m \geq log2.N :\quad r^m_{\langle c,s\rangle} = 1$.
Un estado satisfaciendo $(\exists i : i < log2.N \wedge q)$ *no puede tener una razón de terminación de 1. Esto significa que si* $(\theta_m)_{m\in\mathbb{N}}$ *es una $\langle c,s\rangle$-secuencia en Inv, entonces* θ_m *satisface* $(\exists i : i \geq log2.N \wedge q)$.
Todos los estados θ_m *con* $m \geq log2.N$ *satisfacen* $\mathbb{P}(k=0) = 1$ *, por lo tanto el supremo de la secuencia existe y también satisface* $\mathbb{P}(k=0) = 1$.

A fin de hacer mas legible la prueba, usaremos los siguientes azucares sintácticos:

- $(\exists j : 0 \leq j < K : P_j)\ :=\ (\exists j : 0 \leq j \wedge j < K \wedge P_j)$

- $(\forall j : 0 \leq j < K : P_j) \ := \ (\forall j : 0 \leq j \wedge j < K \ \rightarrow P_j)$
- $(\ \mathbf{N}\ j : 0 \leq j < K : P_j\ ) \geq L \ :=$
  $\big(\exists j_1 : \cdots : \exists j_L : 0 \leq j_1 \wedge j_1 < j_2 \wedge \cdots \wedge j_L < K \ \wedge \ P_{j_1} \wedge \cdots \wedge P_{j_L}\ \big)$
- $\mathsf{prime}.N \ := \ N > 1 \wedge \big(\ \forall i : \ i \geq 1 \wedge N \ \texttt{mod}\ i = 0 \quad \rightarrow \quad i = 1 \vee i = N\ \big)$
- $\mathsf{even}.j \ := \ j \ \texttt{mod}\ 2 = 0$
- $\mathsf{odd}.j \ := \ j \ \texttt{mod}\ 2 = 1$

Por otra parte la notación: $\{\ p\ \}\{\ p'\ \}\ s\ \{\ q\ \}\{\ p'\ \}$ representará que:

- vale la tripla: $\{\ p\ \}\ s\ \{\ q\ \}$
- vale la tripla: $\{\ p'\ \}\ s\ \{\ p'\ \}$, porque el programa s no modifica ninguna variable del predicado p'.
- vale la tripla: $\{\ p \wedge p'\ \}\ s\ \{\ q \wedge p'\ \}$

3.2 Verificación de *FactorTwos*

En primer lugar se define el invariante probabilístico Inv del ciclo. Luego se define un predicado probabilístico q que representa una partición del estado θ que se satisface en ese punto; ya que en una parte vale la condición $(s \ \texttt{mod}\ 2 = 0)$ y en la otra parte vale la condición $(s \ \texttt{mod}\ 2 \neq 0)$. El invariante determinístico I es simplemente un predicado estandard que no es modificado por ninguna variable de *FactorTwos*.

$$\begin{array}{rcl} Inv & := & \mathbb{P}(n-1 = 2^r s \wedge I) = 1 \\ q & := & r_1 + r_2 = 1 \wedge \mathbb{P}(s \ \texttt{mod}\ 2 = 0 \wedge n-1 = 2^r s \wedge I) \geq r_1 \ \wedge \\ & & \mathbb{P}(s \ \texttt{mod}\ 2 \neq 0 \wedge n-1 = 2^r s \wedge I) \geq r_2 \\ I & := & n = N \wedge N \geq 3 \wedge \mathsf{odd}.N \end{array}$$

$\{\ \mathbb{P}(I) = 1\ \}$
$\Rightarrow \{\ \mathbb{P}(n-1 = 2^0(n-1) \wedge I) = 1\ \}$
$r := 0;$
$\{\ \mathbb{P}(n-1 = 2^r(n-1) \wedge I) = 1\ \}$
$s := n-1;$
$\{\ Inv\ \}$
$\Rightarrow \{\ \exists r_1, r_2 : q\ \}$
$\{\ q\ \}$
`while` $(s \ \texttt{mod}\ 2 = 0)$ `do`
    $s := s \ \texttt{div}\ 2;$
    $r := r + 1$
`od`
$\{\ Inv \ \wedge \ \mathbb{P}(s \ \texttt{mod}\ 2 = 0) = 0\ \}$
$\Rightarrow \{\ \mathbb{P}(n-1 = 2^r s \wedge \mathsf{odd}.s \wedge I) = 1\ \}$

En la inicialización se utilizó 3 veces la regla (Assign); y luego se aplicó la regla (Cons), donde para este caso particular: $p' := \mathbb{P}(I) = 1$ y $q' := (\exists r_1, r_2 : q)$. Notar que luego de la inicializacón, se divide el invariante Inv en dos partes disjuntas; en la primera vale la condición $(s \texttt{ mod } 2 = 0)$ con probabilidad de al menos r_1, y en la segunda vale la condición $\neg(s \texttt{ mod } 2 = 0)$ con probabilidad de al menos r_2; por otra parte, la suma total de ambas variables probabilísticas es 1. Luego se usó 2 veces la regla (Exists), donde la premisa es la terna:

$$\{\ q\ \}\ \texttt{if}\ (s \texttt{ mod } 2 = 0)\ \ \texttt{then}\ s := s \texttt{ div } 2;\ r := r + 1\ \texttt{else skip fi}\ \{\ Inv\ \}$$

que se demostrará a continuación. A la salida del ciclo, vale que la variable s es impar con probabilidad 1.

$\{\ q\ \}$
`while` $(s \texttt{ mod } 2 = 0)$ `do`
  $\{\ (s \texttt{ mod } 2 = 0)?q\ \}$
$\Rightarrow \{\ r_1 + r_2 = 1 \wedge \mathbb{P}(n - 1 = 2^r s \wedge s \texttt{ mod } 2 = 0 \wedge I) \geq r_1\ \}$
$\Rightarrow \{\ r_1 + r_2 = 1 \wedge \mathbb{P}(n - 1 = 2^{r+1}(s \texttt{ div } 2) \wedge I) \geq r_1\ \}$
  $s := s \texttt{ div } 2;$
  $\{\ r_1 + r_2 = 1 \wedge \mathbb{P}(n - 1 = 2^{r+1} s \wedge I) \geq r_1\ \}$
  $r := r + 1$
  $\{\ r_1 + r_2 = 1 \wedge \mathbb{P}(n - 1 = 2^r s \wedge I) \geq r_1\ \}$

  $\{\ \neg(s \texttt{ mod } 2 = 0)?q\ \}$
$\Rightarrow \{\ \mathbb{P}(n - 1 = 2^r s \wedge I) \geq r_2\ \}$
  `skip` ;
  $\{\ \mathbb{P}(n - 1 = 2^r s \wedge I) \geq r_2\ \}$
`od`
$\{\ (r_1 + r_2 = 1 \wedge \mathbb{P}(n - 1 = 2^r s \wedge I) \geq r_1)\ +\ (\mathbb{P}(n - 1 = 2^r s \wedge I) \geq r_2)\ \}$
$\Rightarrow \{\ Inv\ \}$

En la rama positiva del `if` se corta el predicado q por la condición $(s \texttt{ mod } 2 = 0)$, con lo cual solo sigue valiendo la expresión correspondiente a la variable probabilística r_1. Notar que el invariante determinístico I asegura que $s > 0$, ya que $N \geq 3$, con lo cual $s = 2(s \texttt{ div } 2)$. Luego se usó dos veces la regla (Assign) y dos veces la regla (Cons). En la rama negativa del `if` se corta el predicado q por la condición $\neg(s \texttt{ mod } 2 = 0)$, aqui fueron usadas las reglas (Skip) y (Cons). Finalmente hay que combinar los efectos de ambas ramas del `if` , esto se logra con el operador +; aplicando la regla (If).

Por lo tanto la siguiente tripla es válida:

$$\{\ \mathbb{P}(I) = 1\ \}$$
$$FactorTwos$$
$$\{\ \mathbb{P}(n - 1 = 2^r s \wedge \mathsf{odd}.s \wedge I) = 1\ \}$$

Si además notamos que los predicados $\mathsf{prime}.N$ y $\neg\mathsf{prime}.N$ no son modificados por ninguna variable de $FactorTwos$, tenemos también la validéz del siguiente par de triplas:

$\{\ \mathbb{P}(I \wedge \mathsf{prime}.N) = 1\ \}$
$FactorTwos$
$\{\ \mathbb{P}(n-1 = 2^r s \wedge \mathsf{odd}.s \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

$\{\ \mathbb{P}(I \wedge \mathsf{prime}.N) = 1\ \}$
$FactorTwos$
$\{\ \mathbb{P}(n-1 = 2^r s \wedge \mathsf{odd}.s \wedge I \wedge \neg\mathsf{prime}.N) = 1\ \}$

3.3 Verificación de $Uniform_a[2, n-1]$

En primer lugar definimos el invariante Inv del ciclo, el cual es una disjunción de dos términos. El primero de ellos q_∞ expresa que en cada una de las $j-$ésimas iteraciones existe una probalidad de que el ciclo termine. El segundo de ellos expresa que desde la primera hasta la i-ésima iteración existe una probabilidad de que el ciclo termine; si ello no ocurrió en las $i-1$ primeras iteraciones, hay una probabilidad de $(1-\rho)^i$ de que se genere un m fuera del rango deseado $[0, N)$ y se itere nuevamente. Notar que I es algún predicado determinístico en el cual vale $n = N$ y no es no es modificado por ninguna de las variables de $Uniform_a[0, n-1]$. Finalmente se define una constante ρ para simplificar la notación.

$$
\begin{array}{rcl}
Inv & := & q_\infty \vee (\exists i : q) \\
q_\infty & := & (\forall j : j > 0 : p) \\
q & := & \mathbb{P}(z = i \wedge \neg(2 \le m < n) \wedge I) = (1-\rho)^i \ \wedge\ (\forall j : 1 \le j \le i : p) \\
p & := & (\ \forall k : 2 \le k < N : \mathbb{P}(z = j \wedge m = k \wedge I) = \frac{\rho}{N-2}(1-\rho)^{j-1}\ ) \\
I & := & n = N \wedge \cdots \\
\rho & := & (N-2)2^{-log2.N}
\end{array}
$$

$\{\ \mathbb{P}(I) = 1\ \}$
$\Rightarrow \{\ \mathbb{P}(0 = 0 \wedge \neg(2 \le n < n) \wedge I) = (1-\rho)^0\ \wedge (\forall j : 1 \le j \le 0 : p)\ \}$
$z := 0;$
$\{\ \mathbb{P}(z = 0 \wedge \neg(2 \le n < n) \wedge I) = (1-\rho)^0\ \wedge (\forall j : 1 \le j \le 0 : p)\ \}$
$m := n;$
$\{\ \mathbb{P}(z = 0 \wedge \neg(2 \le m < n) \wedge I) = (1-\rho)^0\ \wedge (\forall j : 1 \le j \le 0 : p)\ \}$
$\Rightarrow \{\ Inv\ \}$
`while` $\neg(2 \le m < n)$ `do`
    $UnifPow2_n;$
    $z := z + 1$
`od` ;
$\{\ \mathbb{P}(\neg(2 \le m < n)\ ) = 0 \wedge Inv\ \}$
$\Rightarrow \{\ (\forall k : 2 \le k < N : \mathbb{P}(m = k \wedge I) = \frac{1}{N-2})\ \}$
$a := m$
$\{\ (\forall k : 2 \le k < N : \mathbb{P}(a = k \wedge I) = \frac{1}{N-2})\ \}$

En la incialización se usó dos veces la regla (Assign). La primer implicación es inmediata. La segunda implicación se obtiene tomando $i = 0$ en q. Notar que la ultima implicación es válida usando en el predicado q_∞ el siguiente hecho:

$$(1-\rho) \neq 0 \quad \Rightarrow \quad \sum_{j>0}(1-\rho)^{j-1} \;=\; \sum_{j=0}^{\infty}(1-\rho)^{j} \;=\; \frac{1}{1-(1-\rho)} \;=\; \frac{1}{\rho}$$

Para mostrar que Inv es invariante la regla (Or) es usada para dividir la prueba en dos partes, la primera de ellas es simple y la desarrollamos a continuación.

$\{\ q_\infty\ \}$
`while` $\neg(2 \leq m < n)$ `do`
  $\{\neg(2 \leq m < n)?q_\infty\ \}$
$\Rightarrow \{\ \mathbb{P}(true) = 0\ \}$
  $Unifpow2_n$;
  $z := z + 1$
  $\{\ \mathbb{P}(true) = 0\ \}$

  $\{\ (2 \leq m < n)?q_\infty\ \}$
$\Rightarrow \{\ q_\infty\ \}$
  `skip`
  $\{\ q_\infty\ \}$
`od`
$\{\ \mathbb{P}(true) = 0 + q_\infty\ \}$
$\Rightarrow \{\ q_\infty\ \}$
$\Rightarrow \{\ Inv\ \}$

Notar que en q_∞ vale $(2 \leq m < n)$ con probabilidad 1; por lo tanto cuando se lo corta por la condición $\neg(2 \leq m < n)$ se obtiene el predicado $\mathbb{P}(true) = 0$, ya que se está recortando todo el estado probabilístico.

En la rama negativa del `if` , cuando cortamos al predicado q_∞ con la condición $\neg(2 \leq m < n)$, en realidad no se recorta nada porque el mismo considera unicamente aquellos m que están en el rango $[2, N)$.

Finalmente combinamos ambos resultados con el operador probabilístico + y luego se aplica el lema 13. La última implicación se desprende de las definicines de q_∞ y Inv.

En la segunda parte de la prueba, para demostrar el invariante Inv usamos la regla (Exists) aplicada a la terna:

$$\{\ q\ \}\ \texttt{if}\ \neg(0 \leq m < n)\ \texttt{then}\ \ UnifPow2_n;\ z := z + 1\ \texttt{else skip fi}\ \ \{\ Inv\ \}$$

$\{\ \exists i : q\ \}$

$\{\ q\ \}$

`while` $\neg(2 \leq m < n)$ `do`

$\{\neg(2 \leq m < n)?q\ \}$

$\Rightarrow \{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho)^i\ \}$

$\Rightarrow (1-\rho)^i \cdot \quad \{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = 1\ \}$

$\{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I) = 1\ \}\{\mathbb{P}(z = i) = 1\}$

$UnifPow2_n$;

$\{\ (\forall k : 2 \leq k < 2^{-log2.N} : \mathbb{P}(m = k \wedge I) = 2^{-log2.N})\ \}\{\mathbb{P}(z = i) = 1\}$

$\Rightarrow \{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho) \wedge$

$(\ \forall k : 2 \leq k < N : \mathbb{P}(z = i \wedge m = k \wedge I) = \frac{\rho}{N-2}\ )\ \}$

$(1-\rho)^i \cdot \quad \{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho) \wedge$

$(\ \forall k : 2 \leq k < N : \mathbb{P}(z = i \wedge m = k \wedge I) = \frac{\rho}{N-2}\ )\ \}$

$\Rightarrow \{\ \mathbb{P}(z+1 = i+1 \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho)^{i+1} \wedge$

$(\ \forall k : 2 \leq k < N : \mathbb{P}(z+1 = i+1 \wedge m = k \wedge I) = \frac{\rho}{N-2}(1-\rho)^i\ )\ \}$

$z := z+1$

$\{\ \mathbb{P}(z = i+1 \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho)^{i+1} \wedge p[j/i+1]\ \}$

$\{\ (2 \leq m < n)?q\ \}$

$\Rightarrow \{\ (\forall j : 1 \leq j \leq i : p)\ \}$

`skip`

$\{\ (\forall j : 1 \leq j \leq i : p)\ \}$

`od` ;

$\{\ (\ \mathbb{P}(z = i+1 \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho)^{i+1} \wedge p[j/i+1]\ )\ +\ (\forall j : 1 \leq j \leq i : p)\ \}$

$\Rightarrow \{\ q[i/i+1]\ \}$

$\Rightarrow \{\ Inv\ \}$

En la rama positiva del `if` cuando se corta al predicado q con la condición $\neg(2 \leq m < n)$, se descarta el termino derecho de q porque allí se satisface que m está en el rango $[2, N)$. La segunda implicación es una aplicación del lema 10 del apéndice. En la subprueba identada se utilizó que ninguna variable de $UnifPow2_n$ modifica al predicado $z = i$. Por otra parte usando algebra se determina que la probabilidad de que m caiga fuera del en el rango $[2, N)$ es de: $1 - (N-2)2^{-log2.N} = 1 - \rho$, mientras que la probabilidad de que m sea alguna contante particular del rango $[2, N)$ es de: $2^{-log2.N} = \frac{\rho}{N-2}$.

Luego se aplica la regla (Lin $\cdot$) a la terna:

$$\begin{array}{l}\{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = 1\ \} \\ Unif Pow2_n; \\ \{\ \mathbb{P}(z = i \wedge \neg(2 \leq m < n) \wedge I) = (1-\rho) \wedge \\ \ (\ \forall k : 2 \leq k < N : \mathbb{P}(z = i \wedge m = k \wedge I) = \frac{\rho}{N-2}\ )\ \}\end{array}$$

escalando por el valor de la expresión $(1-\rho)^i$. En la implicación posterior al escalado se aplica nuevamente el lema 10 del apéndice; finalmente se aplica la regla (Assign).

Cuando se corta al predicado q por la condición $\neg(2 \leq m < n)$ en la rama negativa del `if` , se descarta el término izquierdo de q; donde se satisface que m está fuera del rango $[2, N)$.

Finalmente se combinan los resultados de las dos ramas del `if` con el operador probabilístico $+$. Notar que en este punto el término $p[j/i+1]$ se agrega a la conjunción $(\forall j : 1 \leq j \leq i : p)$. De esta manera se recupera el predicado q, pero con la variable logica i incrementada en 1. La última implicación es consecuencia de las definciones de q e Inv.

Hasta el momento demostramos que vale la tripla

$$\{\ Inv\ \}\ \texttt{if}\ \neg(2 \leq m < n)\ \texttt{then}\ Unif Pow2_n;\ z := z+1\ \texttt{else skip fi}\ \{\ Inv\ \}$$

Lo que resta comprobar es que el invariante Inv es: $\langle \neg(2 \leq m < n)\ ,\ unifpow2_n; z := z+1 \rangle$-closed

Asumamos alguna interpretación J fija. En primer lugar calculamos la razón de terminación en k-pasos. para los estados que satisfacen q_∞ , la razón de terminación es claramente 1. para un estado satisfaciendo q usamos la tripla de *Hoare* deducida anteriormente.

$$\{\ q\ \}\ \texttt{if}\ \neg(2 \leq m < n)\ \texttt{then}\ Unif Pow2_n;\ z := z+1\ \texttt{else skip fi}\ \{\ q[i/i+1]\ \}$$

Esta da que si $(\theta, J) \models q$ entonces $(\texttt{if}^{\,k}_{\langle c,s\rangle}, J) \models q[i/i+k]$ para cada $k \in \mathbb{N}$. Como $q[i/i+k]$ implica que $\mathbb{P}(2 \leq m < n) \geq 1-(1-\rho)^k$ tenemos

$$(\theta, J) \models q \quad \Rightarrow \quad r(\theta)^k_{\langle c,s\rangle} \geq 1-(1-\rho)^k$$

Entonces para todos los estados que satisfacen el predicado p , la razón de terminación en k-pasos es al menos $1-(1-\rho)^k$, $r^k_{\langle c,s\rangle} \geq 1-(1-\rho)^k$. Posteriormente mostramos que alguna $\langle c,s\rangle$-secuencia dentro de p termina en un estado satisfaciendo q_∞ , para ello el predicado Inv es separado en dos partes:

$$\begin{array}{rrcl} & Inv & \equiv & p_k \vee (\exists i : i < k \wedge q) \\ \text{donde:} & p_k & := & q_\infty \vee (\exists i : i \geq k \wedge q)\end{array}$$

Un estado satisfaciendo $(\exists i : i < k \wedge q)$ no puede tener una razón de terminación de $1-(1-\rho)^k$. Esto significa que si $(\theta_k)_{k\in\mathbb{N}}$ es una $\langle c,s\rangle$-secuencia dentro de p entonces θ_n satisface p_k . Todos los estados θ_k con $k \geq j$ satisfacen $\mathbb{P}(2 \leq m < n \wedge z = j) = \rho(1-\rho)^{j-1}$. La probabilidad $\mathbb{P}(2 \leq m < n \wedge z = j)$ no puede cambiar en el supremo de la secuencia. La secuencia debe entonces terminar en un estado satisfaciendo $\mathbb{P}(2 \leq m < n \wedge z = j) = \rho(1-\rho)^{j-1}$. Esto vale para todos los valores de j . Notar que el predicado Inv no sería cerrado sin la presencia del termino q_∞ .

De esta manera se obtiene el generador de numeros aleatorios necesario para ejecutar el algoritmo de $MillerRabin$. Lo llamaremos $Uniform_a[2, n-1]$, el mismo almacenará en la variable a un numero con distribución uniforme en el rango $[2, n-1]$. Notar que la precondición y postcondición de la tripla verificada, es independiente de la variable z; es decir que dicha variable fue utilizada para demostrar que Inv es invariante, llevando la cuenta de la cantidad de veces que se iteró. Por lo tanto podemos eliminar z, ya que no participa en el control de flujo del programa.

$$
\begin{array}{l}
\{\ \mathbb{P}(I) = 1\ \} \\
m := n; \\
\texttt{while}\ \neg(2 \leq m < n)\ \texttt{do} \\
\qquad UnifPow2_n \\
\texttt{od}\ ; \\
a := m \\
\{\ (\forall k : 2 \leq k < N : \mathbb{P}(a = k \wedge I) = \tfrac{1}{N-2})\ \}
\end{array}
$$

3.4 Verificación de $WitnessTail$

Primeramente definimos el invariante Inv del ciclo.

$$
\begin{array}{rcl}
Inv & := & \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) = 1 \\
w(e) & := & (\exists j : 0 \leq j < e : a^{2^{j+1} s} \ \texttt{mod}\ n = 1 \wedge a^{2^j s} \ \texttt{mod}\ n \neq 1 \wedge a^{2^j s} \ \texttt{mod}\ n = n-1) \\
I & := & n-1 = 2^r s \ \wedge\ \texttt{odd}.s \ \wedge\ n = N \ \wedge\ N \geq 3 \ \wedge\ \texttt{odd}.N \\
q_1 & := & r_1 + r_2 = 1 \ \wedge \\
 & & \mathbb{P}\big(\ (y < r \wedge \neg witness) \ \wedge\ y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \ \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_1 \ \wedge \\
 & & \mathbb{P}\big(\neg(y < r \wedge \neg witness) \ \wedge\ y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \ \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_2
\end{array}
$$

Notar que en la variable a' se va calculando la siguiente secuencia:

$$\big(a^{2^0 s} \ \texttt{mod}\ n, \ldots, a^{2^r s} \ \texttt{mod}\ n\big)$$

Si alguno de los elementos de la secuencia no satisface algunos de los test de primalidad esperados (los del primer capítulo); a se convierte en una base testigo y ello queda registrado en la variable booleana $witness$. El predicado $w(e)$ expresa la parte de la secuencia sobre la cual ya se ha verificado los test de primalidad:

$$\big(a^{2^0 s} \ \texttt{mod}\ n, \ldots, a^{2^{e-1} s} \ \texttt{mod}\ n\big)$$

Es fácil ver que por definición se satisfacen las siguientes equivalencias:

$$
\begin{array}{rcl}
w(0) & \Leftrightarrow & false \\
w(y+1) & \Leftrightarrow & w(y) \vee (a^{2^{y+1} s} \ \texttt{mod}\ n = 1 \wedge a^{2^y s} \ \texttt{mod}\ n \neq 1 \wedge a^{2^y s} \ \texttt{mod}\ n = n-1) \\
w(r) \vee (a^{n-1} \ \texttt{mod}\ n \neq 1) & \Leftrightarrow & \texttt{witness}.n.a
\end{array}
$$

Igual que en las verifiaciones anteriores, el invariante estandard I tampoco es modificado por alguna variable del programa $WitnessTail$.

$\{\ \mathbb{P}(I) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}\big(0 \leq r \wedge a^s \ \texttt{mod}\ n = a^{2^0 s} \ \texttt{mod}\ n\ \wedge (false \Leftrightarrow w(0)) \wedge I\big) = 1\ \}$

$a' := a^s \ \texttt{mod}\ n;$

$\{\ \mathbb{P}\big(0 \leq r \wedge a' = a^{2^0 s} \ \texttt{mod}\ n\ \wedge (false \Leftrightarrow w(0)) \wedge I\big) = 1\ \}$

$y := 0;$

$\{\ \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n\ \wedge (false \Leftrightarrow w(y)) \wedge I\big) = 1\ \}$

$witness := false;$

$\{\ Inv\ \}$

$\Rightarrow \{\ \exists r_1, r_2 : q_1\ \}$

$\{\ q_1\ \}$

$\texttt{while}\ (y < r \wedge \neg witness)\ \texttt{do}$

$\quad \texttt{if}\ (a' \neq 1 \wedge a' \neq n-1)\ \texttt{then}$

$\quad\quad a' := a'^2 \ \texttt{mod}\ n;$

$\quad\quad witness := (a' = 1)$

$\quad \texttt{else}$

$\quad\quad a' := a'^2 \ \texttt{mod}\ n$

$\quad \texttt{fi}\ ;$

$\quad y := y + 1$

$\texttt{od}\ ;$

$\{\ \mathbb{P}(y < r \wedge \neg witness) = 0\ \wedge\ Inv\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\neg(y < r \wedge \neg witness) \wedge y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\ \big(y \geq r \wedge y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \vee$

$\quad \big(witness \wedge y \leq r \wedge a' = a^{2^y s} \ \texttt{mod}\ n \wedge (witness \Leftrightarrow w(y)) \wedge I\big)\ \big) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\ (a' = a^{n-1} \ \texttt{mod}\ n \wedge (witness \Leftrightarrow w(r)) \wedge I) \vee$

$\quad ((witness \vee a' \neq 1 \Leftrightarrow w(r) \vee a^{n-1} \ \texttt{mod}\ n \neq 1) \wedge I)\ \big) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\ (witness \vee a' \neq 1 \Leftrightarrow w(r) \vee a^{n-1} \ \texttt{mod}\ n \neq 1) \wedge I\ \big) = 1\ \}$

$witness := witness \vee a' \neq 1;$

$\{\ \mathbb{P}\big(\ (witness \Leftrightarrow w(r) \vee a^{n-1} \ \texttt{mod}\ n \neq 1) \wedge I\ \big) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}((witness \Leftrightarrow \mathsf{witness}.n.a) \wedge I) = 1\ \}$

En la parte de inicialización se usó tres veces la regla (Assign). Luego se aplicó la regla (Exists) donde la premisa correspondiente es la tripla:

$$\{\ q_1\ \}\ \texttt{if}\ (y < r \wedge \neg witness)\ \texttt{then}\ \cdots\ \texttt{else skip fi}\ \{\ Inv\ \}$$

la cual se demostrará luego. En la salida del ciclo, como vale la condición $(y < r \wedge \neg witness)$ con probabilidad 0, y el invariante implica que la probabilidad total es 1, se tiene que la probabilidad de $\neg(y < r \wedge \neg witness)$ es 1; justificándose de esta manera la primer implicación. En la segunda implicación se utiliza que $\wedge$ se distribuye con respecto a $\vee$ quedando así una disyunción de dos términos. En la tercer implicación, en el término izquierdo se usó que $y = r$ y $2^r s = n - 1$ (asegurado por I); mientras que en el término derecho se recurrió a la siguiente propiedad lógica:

$$(A \wedge (A \Leftrightarrow B)) \quad \Rightarrow \quad ((A \vee A') \Leftrightarrow (B \vee B'))$$

La cuarta implicación de la salida del ciclo es por el siguiente hecho:

$$(A \Rightarrow B) \quad \Rightarrow \quad (A \vee B) \Rightarrow B$$

y la última es por definición de witness.$n.a$.

A continuación se demuestra la tripla pendiente, antes de ello definimos el siguiente predicado que será necesario para verificar el `if` anidado en el ciclo.

$$\begin{aligned} q_2 \quad := \quad & r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge \\ & \mathbb{P}\big(\ (a' \neq 1 \wedge a' \neq n-1) \wedge y < r \wedge \neg witness \wedge a' = a^{2^y s} \bmod n \wedge \\ & \quad (witness \Leftrightarrow w(y))\ \big) \geq r_3 \wedge \\ & \mathbb{P}\big(\neg(a' \neq 1 \wedge a' \neq n-1) \wedge y < r \wedge \neg witness \wedge a' = a^{2^y s} \bmod n \wedge \\ & \quad (witness \Leftrightarrow w(y))\ \big) \geq r_4 \end{aligned}$$

$\{\ q_1\ \}$
`while` $(y < r \wedge \neg witness)$ `do`
  $\{\ (y < r \wedge \neg witness)?q_1\ \}$
$\Rightarrow \{\ r_1 + r_2 = 1 \wedge$
  $\mathbb{P}\big(\ y < r \wedge \neg witness \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_1\ \}$
$\Rightarrow \{\ \exists r_3, r_4 : q_2\ \}$
  $\{\ q_2\ \}$
  `if` $(a' \neq 1 \wedge a' \neq n-1)$ `then`
    $a' := a'^2 \bmod n;$
    $witness := a' = 1$
  `else`
    $a' := a'^2 \bmod n$
  `fi` ;
  $\{\ r_1 + r_2 = 1 \wedge \mathbb{P}\big(y + 1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_1\ \}$
  $y := y + 1$
  $\{\ r_1 + r_2 = 1 \wedge \mathbb{P}\big(\ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_1\ \}$

  $\{\neg(y < r \wedge \neg witness)?q_1\ \}$
$\Rightarrow \{\ \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_2\ \}$
  `skip`
  $\{\ \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_2\ \}$
`od`
$\{\big(\ r_1 + r_2 = 1 \wedge\ \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_1\ \big) +$
$\big(\ \mathbb{P}\big(y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_2\ \big)\ \}$
$\Rightarrow \{\ Inv\ \}$

Notar que aquí todas las implicaciones son aplicaciones de los lemas del apéndice. En la rama positiva del `if` se usó en primer lugar la regla (Exists) y en segundo lugar la regla (Assign). Mientras que en la rama negativa del `if` se aplicó la regla (Skip).

$\{\ q_2\ \}$

`if` $(a' \neq 1 \wedge a' \neq n-1)$ `then`

$\quad \{\ (a' \neq 1 \wedge a' \neq n-1)?q_2\ \}$

$\Rightarrow \{\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ (a' \neq 1 \wedge a' \neq n-1) \wedge y < r \wedge \neg witness \wedge a' = a^{2^y s} \bmod n \wedge$

$\quad\quad (witness \Leftrightarrow w(y)) \wedge I\big) \geq r_3\ \}$

$\Rightarrow \{\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ (a' \neq 1 \wedge a' \neq n-1) \wedge y < r \wedge \neg witness \wedge a' = a^{2^y s} \bmod n \wedge$

$\quad\quad (\ witness \vee (a'^2 \bmod n = 1 \wedge a' \neq 1 \wedge a' \neq n-1) \Leftrightarrow$

$\quad\quad\quad w(y) \vee (a^{2^{y+1} s} \bmod n = 1 \wedge a^{2^y s} \bmod n \neq 1 \wedge a^{2^y s} \bmod n = n-1)\ ) \wedge I\big) \geq r_3\ \}$

$\Rightarrow \{\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n\ \wedge (a'^2 \bmod n = 1 \Leftrightarrow w(y+1)) \wedge I\big) \geq r_3\ \}$

$\quad a' := a'^2 \bmod n;$

$\quad \{\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (a' = 1 \Leftrightarrow w(y+1)) \wedge I\big) \geq r_3\ \}$

$\quad witness := a' = 1$

$\quad \{\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_3\ \}$

`else`

$\quad \{\neg(a' \neq 1 \wedge a' \neq n-1)?q_2\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\neg(a' \neq 1 \wedge a' \neq n-1) \wedge y < r \wedge \neg witness\ \wedge a' = a^{2^y s} \bmod n \wedge$

$\quad (witness \vee (a'^2 \bmod n = 1 \wedge a' \neq 1 \wedge a' \neq n-1) \Leftrightarrow w(y+1)) \wedge I\big) \geq r_4\ \}$

$\Rightarrow \{\ \mathbb{P}\big(\ y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_4\ \}$

$\quad a' := a'^2 \bmod n$

$\quad \{\ \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_4\ \}$

`fi`

$\{\ \big(\ r_3 + r_4 = r_1 \wedge r_1 + r_2 = 1 \wedge$

$\quad \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_3\ \big)\ +$

$\big(\ \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_4\ \big)\ \}$

$\Rightarrow \{\ r_1 + r_2 = 1 \wedge\ \mathbb{P}\big(\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n\ \wedge (witness \Leftrightarrow w(y+1)) \wedge I\big) \geq r_1\ \}$

3.5 Verificación de $MillerRabin1$, con N primo

Se define el siguiente invariante estandard I que ninguna variable de $MillerRabin1$ modifica.

$$I \ := \ n-1 = 2^r s \ \wedge\ \mathtt{odd}.s \ \wedge\ n = N \ \wedge\ N \geq 3 \ \wedge\ \mathtt{odd}.N$$

$\{\ \mathbb{P}(I \wedge \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I) = 1\ \}\{\ \mathbb{P}(\mathsf{prime}.N) = 1\ \}$

$Uniform_a[2, n-1];$

$\{\ (\forall j : 2 \leq j \leq N-1 : \mathbb{P}(a = j \wedge I) = \frac{1}{N-2})\ \}\{\ \mathbb{P}(\mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I \wedge 0 < a < n \wedge \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I) = 1\ \}\{\ \mathbb{P}(\neg\mathsf{witness}.n.a \wedge \mathsf{prime}.N) = 1\ \}$

$WitnessTail$

$\{\ \mathbb{P}((witness \Leftrightarrow \mathsf{witness}.n.a) \wedge I) = 1\}\{\ \mathbb{P}(\neg\mathsf{witness}.n.a \wedge \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(\neg witness \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

La primer implicación es inmediata y la notación $\{\ p\ \}\{\ p'\ \}$ es una manera conveniente de representar la conjunción entre ambos predicados: $\{\ p \wedge p'\ \}$. El término derecho de la conjunción $\mathbb{P}(\mathsf{prime}.N) = 1$ no es modificado por ninguna variable de $Uniform_a[2, n-1]$. La segunda implicación se obtiene sumando cada una de las $N-2$ probabilidades de que a sea alguna constante en el rango $[2, N-1]$. La tercer implicación es la más importante, y es aquí donde se aplica una de las propiedades matemáticas fundamentales enunciadas al comienzo:

$$0 < a < n \wedge \mathsf{prime}.n \quad \Rightarrow \quad \neg\mathsf{witness}.n.a$$

De manera similar, el predicado: $\mathbb{P}(\neg\mathsf{witness}.n.a \wedge \mathsf{prime}.N) = 1$
no es alterado por ninguna variable del programa $WitnessTail$. En la última implicación se usó la siguiente propiedad lógica:

$$((A \Leftrightarrow B) \wedge \neg B) \quad \Rightarrow \quad \neg A$$

Notar que la propiedad que brinda esta verficación es muy importante, porque significa que cualquiera sea la base a sorteada en el rango $[2, N-1]$, nunca será un testigo; es decir: $\mathbb{P}(\neg witness) = 1$. La repercusión que tiene esto es grande, porque en cada una de las T iteraciones que ejecute el programa principal $MllerRabin$, nunca encontrará un testigo, es decir que valdrá: $\mathbb{P}(prime) = 1$.

3.6 Verificación de $MillerRabin1$, con N compuesto

Definimos el mismo invariante estandard que antes:

$$I \quad := \quad n-1 = 2^r s \ \wedge\ \mathsf{odd}.s \ \wedge\ n = N \ \wedge\ N \geq 3 \ \wedge\ \mathsf{odd}.N$$

$\{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I) = 1\ \}\{\ (\mathbf{N}j : 0 < j < N : \mathsf{witness}.N.j) \geq \frac{N-1}{2}\ \}$

$Uniform_a[2, n-1];$

$\{\ (\forall j : 2 \leq j \leq N-1 : \mathbb{P}(a = j \wedge I) = \frac{1}{N-2})\ \}\{\ (\mathbf{N}j : 0 < j < N : \mathsf{witness}.N.j) \geq \frac{N-1}{2}\ \}$

$\Rightarrow \{\ \mathbb{P}(I) = 1\ \}\{\ \mathbb{P}(\mathsf{witness}.n.a) \geq \frac{1}{2}\ \}$

$WitnessTail$

$\{\ \mathbb{P}((witness \Leftrightarrow \mathsf{witness}.n.a) \wedge I) = 1\ \}\{\ \mathbb{P}(\mathsf{witness}.n.a) \geq \frac{1}{2}\ \}$

$\Rightarrow \{\ \mathbb{P}(\neg witness) \geq \frac{1}{2}\ \wedge\ \mathbb{P}(true) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(\neg witness) \leq \frac{1}{2}\ \}$

La primer implicación es la más importante y es aquí donde se aplica la otra propiedad matemática fundamental enunciada al comienzo:

$$n \geq 3 \wedge \mathsf{odd.n} \wedge \neg \mathsf{prime}.n \quad \Rightarrow \quad |\ \{a\ :\ 0 < a < n \wedge \mathsf{witness}.n.a\ \}\ | \geq \frac{n-1}{2}$$

En la segunda implicación, $\mathbb{P}(I) = 1$ se recupera sumando cada una de las $N-2$ probabilidades de que a sea una constante del rango $[2, N-1]$. Por otra parte como existen al menos $\frac{N-1}{2}$ bases j que son testigos, y la probabilidad de que a sea cualquiera de ellas es de $\frac{1}{N-2}$ entonces la probabilidad de que a sea un testigo será: $(\frac{N-1}{2})(\frac{1}{N-2}) \geq \frac{1}{2}$. Por lo tanto el predicado $\mathbb{P}(\mathsf{witness}.n.a) \geq \frac{1}{2}$ también vale en la segunda implicación. En la penúltima implicación, como la probabilidad que valga $\mathsf{witness}.n.a$ es de al menos $\frac{1}{2}$ y por otra parte en todos los estados determinísticos del estado probabilístico se satisface la equivalencia: $witness \Leftrightarrow \mathsf{witness}.n.a$; entonces se tiene que la probabilidad de que $witness$ sea verdadero es al menos: $\frac{1}{2}$. La última implicación es consecuencia de que la probabilidad total es 1 y por lo tanto la probabilidad de que valga $\neg witness$ es a lo sumo $1 - \frac{1}{2} = \frac{1}{2}$.

También en esta caso (N compuesto), la repercusión de la validéz de la terna:

$$\{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) = 1\ \}\ MillerRabin1\ \{\ \mathbb{P}(\neg witness) \leq \frac{1}{2}\ \}$$

es muy grande, ya que si se ejecutan T iteraciones de este fragmento en el programa principal $MillerRabin$; la probabilidad de que en cada una de ellas $witness$ sea falso será $\leq \frac{1}{2} \cdots \frac{1}{2} = \frac{1}{2}^T$. Es decir que la probabilidad de que no aparezcan testigos decrece con cada iteración: $\mathbb{P}(\neg prime) \leq \frac{1}{2}^T$.

3.7 Verificación de $MillerRabin$, con N primo

En primera instancia definimos el invariante Inv del ciclo.

$$\begin{array}{rcl} Inv & := & p \vee (\exists i : q) \\ p & := & \mathbb{P}(prime \wedge x = T \wedge I \wedge \mathsf{prime}.N) = 1 \\ q & := & i < T \wedge \mathbb{P}(prime \wedge x = i \wedge I \wedge \mathsf{prime}.N) = 1 \\ I & := & n-1 = 2^r s\ \wedge\ \mathsf{odd}.s\ \wedge\ n = N\ \wedge\ N \geq 3\ \wedge\ \mathsf{odd}.N \end{array}$$

El invariante expresa que siempre vale $prime$ con probabilidad 1. La presencia del término $\mathsf{prime}.N$ es fundamental para que en cada iteración la probabilidad de $prime$ se mantenga igual a 1. Observar que o bien se está en la ultima iteracion T, o existe un $i < T$ tal que se está en la i-ésima iteración. De manera

similar a las verifiaciones anteriores, I es un invariante estandard que no es modificado por ninguna varible del programa $MillerRabin$.

$\{\ \mathbb{P}(I \wedge \mathsf{prime}.N) = 1\ \}$
$\Rightarrow \{\ \mathbb{P}(true \wedge 0 = 0 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$
$prime := true;$
$\{\ \mathbb{P}(prime \wedge 0 = 0 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$
$x := 0;$
$\{\ \mathbb{P}(prime \wedge x = 0 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$
$\Rightarrow \{\ Inv\ \}$
`while` $(x < T)$ `do`
    $MillerRabin1;$
    $prime := prime \wedge \neg witness;$
    $x := x + 1$
`od`
$\{\ \mathbb{P}(x < T) = 0 \wedge Inv\ \}$
$\Rightarrow \{\ \mathbb{P}(prime \wedge x = T \wedge I \wedge \mathsf{prime}.N) = 1\ \}$
$\Rightarrow \{\ \mathbb{P}(prime) = 1\ \}$

En la inicialización se aplicó dos veces la regla (Assign) y luego la regla (Cons). En la salida del ciclo vale $x < T$ con probabilidad 0; con lo cual la parte del invariante que sobrevive es p, ya que algún valor $i < T$ implica una probabilidad positiva para $x < T$. A fin de mostrar que Inv es invariante la regla (Or) es usada para dividir la prueba en dos partes, una donde la precondición es $\{\ \exists i : q\ \}$ y otra donde la precondición es $\{\ p\ \}$.

$\{\ \exists i : q\ \}$

$\Rightarrow \{\ q\ \}$

`while` $(x < T)$ `do`

$\{\ (x < T)?q\ \}$

$\Rightarrow \{\ i < T \wedge \mathbb{P}(prime \wedge x = i \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I \wedge \mathsf{prime}.N) = 1\ \}\{\ i < T \wedge \mathbb{P}(prime \wedge x = i) = 1\ \}$

$MillerRabin1$;

$\{\ \mathbb{P}(\neg witness \wedge I \wedge \mathsf{prime}.N) = 1\ \}\{\ i < T \wedge \mathbb{P}(prime \wedge x = i) = 1\ \}$

$\Rightarrow \{\ i{+}1 \leq T \wedge \mathbb{P}(prime \wedge \neg witness \wedge x + 1 = i{+}1 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

$prime := prime \wedge \neg witness$;

$\{\ i{+}1 \leq T \wedge \mathbb{P}(prime \wedge x + 1 = i{+}1 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

$x := x + 1$

$\{\ i{+}1 \leq T \wedge \mathbb{P}(prime \wedge x = i{+}1 \wedge I \wedge \mathsf{prime}.N) = 1\ \}$

$\{\neg(x < T)?q\ \}$

`skip`

$\{\ true\ \}$

`od`

$\{\ (\ i{+}1 < T \wedge \mathbb{P}(prime \wedge x = i{+}1 \wedge I \wedge \mathsf{prime}.N) = 1\ )\ +\ true\ \}$

$\Rightarrow \{\ q[i/i{+}1]\ \}$

$\Rightarrow \{\ Inv\ \}$

Aquí se aplicó la regla (Exists) a la tripla:

$$\{\ q\ \}\ \texttt{if}\ (x < T)\ \texttt{then}\ \cdots\ \texttt{else skip fi}\ \{\ Inv\ \}$$

En la rama positiva del `if` cuando se corta a q por la condición $x < T$, en definitiva no se recorta nada, porque en el predicado q vale $x = i$ con probabilidad 1, pero $i < T$. La segunda implicación se desprende de que todos los predicados determinísticos dentro del constructor $\mathbb{P}(\cdots)$ valen con probabilidad 1. Nuevamente la notación $\{\ p\ \}\{\ p'\ \}$ representa la conjunción $\{\ p \wedge p'\ \}$. Seguidamente la tripla de la izquierda es la verificada anteriormente, mientras que la tripla de la derecha:

$$\{\ i < T \wedge \mathbb{P}(prime \wedge x = i) = 1\ \}\ MillerRabin1\ \{\ i < T \wedge \mathbb{P}(prime \wedge x = i) = 1\ \}$$

es válida porque ninguna variable del programa $MillerRabin1$ modifica las variables $prime, x$. La tercer implicación se satisface por las mismas razones que la primer implicación. Finalmente se aplica dos veces la regla (Assign).

En la rama negativa del `if` , solo se hizo uso que $true$ se satisface en todas partes.

Cuando se combinan ambas partes del `if` con el operador probabilístico $+$, la probabilidad $= 1$ se convierte en ≥ 1; pero como la probabilidad total siempre es ≤ 1, se mantiene el $= 1$. Notar que en la rama negativa del `if` , cuando se corta a q por la condición $\neg(x < T)$, en lugar de haber puesto $true$ se podría haber puesto:

$$\mathbb{P}(prime \wedge x = i \wedge I \wedge \mathsf{prime}.N) = 0$$

Con lo cual, cuando se combinan ambas partes del `if` con el operador + se mantiene el $= 1$, de una forma más precisa. La última impliación es inmediata, $q[i/i+1]$ representa el predicado q, donde se hace el reemplazo sintáctico $i \leftarrow i+1$.

$\{\ p\ \}$
`while` $(x < T)$ `do`
$\{\ (x < T)?p\ \}$
$MillerRabin1$;
$prime := prime \wedge \neg witness$;
$x := x + 1$
$\{\ true\ \}$

$\{\neg(x < T)?p\ \}$
$\Rightarrow \{\ p\ \}$
`skip`
$\{\ p\ \}$
`od`
$\{\ true + p\ \}$
$\Rightarrow \{\ p\ \}$
$\Rightarrow \{\ Inv\ \}$

Para este caso pueden hacerce observaciones similares a las de la verificación anterior. El predicado $true$ se satisface en cualquier parte. Cuando se recorta a p por la condición $\neg(x < T)$ la probabilidad en p no decrece porque vale $x = T$ con probabilidad 1. Cuando se combinan ambas partes de `if` con el operador +, también aquí se conserva el $= 1$.

3.8 Verificación de $MillerRabin$, con N compuesto

Se define el invariante Inv del ciclo:

$$\begin{array}{rcl} Inv & := & p \vee (\exists i : q) \\ p & := & \mathbb{P}(prime) \leq \frac{1}{2}^T \wedge \mathbb{P}(x = T \wedge I \wedge \neg \mathsf{prime}.N) = 1 \\ q & := & i < T \wedge \mathbb{P}(prime) \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1 \\ I & := & n-1 = 2^r s \ \wedge\ \mathsf{odd}.s \ \wedge\ n = N \ \wedge\ N \geq 3 \ \wedge\ \mathsf{odd}.N \end{array}$$

El invariante es una disjunción de dos términos. El término p expresa que la probabilidad de estar en la última iteración T es 1 y que la probabilidad de que $prime$ sea verdadero en la última iteración esta acotada superiormente por $\frac{1}{2}^T$. El término $(\exists i : q)$ expresa que existe un $i < T$ tal que se está en la i-ésima iteración con probabilidad 1 y que la probabilidad de que $prime$ sea verdadero en ese punto es a lo sumo $\frac{1}{2}^i$. Notar que el predicado $\neg\mathsf{prime}.N$ es fundamental aquí para garantizar que en cada una de las iteraciones, la probabilidad de que aparezca una base a testigo en $MillerRabin1$ sea a lo sumo $\frac{1}{2}$. Igual que en las verificaciones anteriores el invariante estandard I no es modificado por ninguna variable

de $MillerRabin$.

$$
\begin{array}{l}
\quad \{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) = 1\ \} \\
\Rightarrow \{\ \mathbb{P}(true) \leq \frac{1}{2}^0 \wedge \mathbb{P}(0 = 0 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \} \\
\quad prime := true; \\
\quad \{\ \mathbb{P}(prime) \leq \frac{1}{2}^0 \wedge \mathbb{P}(0 = 0 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \} \\
\quad x := 0; \\
\quad \{\ \mathbb{P}(prime) \leq \frac{1}{2}^0 \wedge \mathbb{P}(x = 0 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \} \\
\Rightarrow \{\ Inv\ \} \\
\quad \texttt{while}\ (x < T)\ \texttt{do} \\
\qquad MillerRabin1; \\
\qquad prime := prime \wedge \neg witness; \\
\qquad x := x + 1 \\
\quad \texttt{od} \\
\quad \{\ \mathbb{P}(x < T) = 0 \wedge Inv\ \} \\
\Rightarrow \{\ \mathbb{P}(prime) \leq \frac{1}{2}^T \wedge \mathbb{P}(x = T \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \} \\
\Rightarrow \{\ \mathbb{P}(\neg prime) \geq 1 - \frac{1}{2}^T\ \}
\end{array}
$$

En la parte de inicialización se aplicó dos veces la regla (Assign) y luego la regla (Cons). La segunda implicación se justifica tomando $i = 0$ en el predicado q de Inv. En la salida del ciclo se tiene que vale p ; ya que algún valor $i < T$ se tiene una probabilidad positiva para $x < T$. Es importante destacar que en la última implicación, la presencia del término $\mathbb{P}(x = T \wedge I \wedge \neg \mathsf{prime}.N) = 1$ es fundamental. La importancia radica en que el mismo implica que la probabilidad total es 1, lo cual permite hablar acerca de la probabilidad de $\neg prime$ en función de la probabilidad de $prime$. Igual que antes a fin de probar que Inv es invariante se utiliza la regla (Or) para dividir la prueba en dos partes: una donde la precondición es $\{\ \exists i : q\ \}$ y otra donde donde la precondición es $\{\ p\ \}$.

$\{\ \exists i : q\ \}$

$\{\ q\ \}$

`while` $(x < T)$ `do`

$\quad \{\ (x < T)?q\ \}$

$\Rightarrow \{\ \exists r : \mathbb{P}(prime) = r \wedge i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\}$

$\quad \{\ \mathbb{P}(prime) = r \wedge i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq r \wedge i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\}$

$\Rightarrow \{\ r \cdot \mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq 1\ \}\ \{i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\quad \{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq 1\ \}$

$\quad \Rightarrow \{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\quad MillerRabin1;$

$\quad \{\ \mathbb{P}(\neg witness) \leq \frac{1}{2}\ \}$

$\quad \Rightarrow \{\ \mathbb{P}(prime \wedge \neg witness) \leq \frac{1}{2}\ \}$

$\quad prime := prime \wedge \neg witness;$

$\quad \{\ \mathbb{P}(prime) \leq \frac{1}{2}\ \}$

$\{\ r \cdot \mathbb{P}(prime) \leq \frac{1}{2}\ \}\ \{\ i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\Rightarrow \{\ i < T \wedge \mathbb{P}(prime) \leq \frac{1}{2}^{i+1} \wedge \mathbb{P}(x + 1 = i + 1 \wedge I \wedge \neg \mathsf{prime}.N) = 1\}$

$x := x + 1$

$\{\ i < T \wedge \mathbb{P}(prime) \leq \frac{1}{2}^{i+1} \wedge \mathbb{P}(x = i + 1 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$\{\neg(x < T)?q\ \}$

$\Rightarrow \{\ \mathbb{P}(true) = 0\ \}$

`skip`

$\{\ \mathbb{P}(true) = 0\ \}$

`od`

$\{\ (\ i < T \wedge \mathbb{P}(prime) \leq \frac{1}{2}^{i+1} \wedge \mathbb{P}(x = i + 1 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ )\ +\ (\ \mathbb{P}(true) = 0\ )\ \}$

$\Rightarrow \{\ q[i/i{+}1]\ \}$

$\Rightarrow \{\ Inv\ \}$

Notar que se aplica la regla (Exists) a la tripla:

$$\{\ q\ \}\ \texttt{if}\ (x < T)\ \texttt{then}\ MillerRabin1; prime := prime \wedge \neg witness; x := x + 1\ \texttt{else skip fi}\ \{\ Inv\ \}$$

En la rama positiva del `if` , la primer implicación introduce una variable probabilística r que representa la probabilidad exacta de que valga el predicado *prime*. Notar que se aplica la regla (Exists) a la tripla:

$\{\ \mathbb{P}(prime) = r \wedge i < T \wedge r \leq \frac{1}{2}^i \wedge \mathbb{P}(x = i \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

$MillerRabin1;$

$prime := prime \wedge \neg witness;$

$x := x + 1$

$\{\ i < T \wedge \mathbb{P}(prime) \leq \frac{1}{2}^{i+1} \wedge \mathbb{P}(x = i + 1 \wedge I \wedge \neg \mathsf{prime}.N) = 1\ \}$

En la segunda implicación vale el predicado $(I \wedge \neg \mathsf{prime}.N)$ con probabilidad $= 1 \geq \frac{1}{2}^i \geq r$, con lo cual se obtiene el término izquierdo de la conjunción. Teniendo que en cuenta que la variable probabilística r toma un valor en rango $\left(0, \frac{1}{2}^i\right)$, en la tercer implicación se aplicó el lema 10 del apéndice: el predicado $\mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq r$ es equivalente al predicado $r \cdot (\mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq 1)$. La cuarta implicación es inmediata, ya que la probabilidad total no puede exceder a 1. En la quinta impliación se usó el lema 6; debilitando el predicado $(\neg witness)$ a $(prime \wedge \neg witness)$ se preserva el $\leq$. En la última implicación de la rama positiva del `if` también se usó el lema 10 del apéndice: el predicado $r \cdot (\mathbb{P}(prime) \leq \frac{1}{2})$ es equivalente al predicado $\mathbb{P}(prime) \leq r \cdot \frac{1}{2}$; con lo cual la probabilidad de que valga el predicado $prime$ es $\leq r \cdot \frac{1}{2} \leq \left(\frac{1}{2}^i\right)\frac{1}{2} = \frac{1}{2}^{i+1}$. Notar que cuando se aplca la regla (Lin $\cdot$) a la tripla:

$$
\begin{array}{l}
\{\ \mathbb{P}(I \wedge \neg \mathsf{prime}.N) \geq 1\ \} \\
MillerRabin1; \\
prime := prime \wedge \neg witness \\
\{\ \mathbb{P}(prime) \leq \frac{1}{2}\ \}
\end{array}
$$

se está escalando por el valor de la variable probabilística r, el cual se encuentra en el rango $\left(0, \frac{1}{2}^i\right)$.

En la rama negativa del `if` , cuando se corta al predicado q por la condición $\neg(x < T)$, se obtiene el predicado $\mathbb{P}(true) = 0$. Es importante destacar que el único estado probabilístico que satisface este predicado es θ_0, ello se debe a que en q se satisface $x < T$ con probabilidad 1. Finalmente se usó la regla (Skip).

Luego se combinan los efectos que realizaron ambas ramas positiva y negativa del `if` sobre los predicados $(x < T)?q$ y $\neg(x < T)?q$ respectivamente, con el operador probabilístico $+$. Usando el lema 13 del apéndice las probabilidades en el término izquierdo de la suma son preservadas, ya que el estado θ_0 no agrega ninguna probabilidad. Las dos últimas implicaciones son consecuencia inmediata de las definciones de q y Inv.

$\{\ p\ \}$

`while` $(x < T)$ `do`

$\quad \{\ (x < T)?p\ \}$

$\Rightarrow \{\ \mathbb{P}(true) = 0\ \}$

$\quad MillerRabin1;$

$\quad prime := prime \wedge \neg witness;$

$\quad x := x + 1$

$\quad \{\ \mathbb{P}(true) = 0\ \}$

$\quad \_\_\_\_\_$

$\quad \{\neg(x < T)?p\ \}$

$\Rightarrow \{\ p\ \}$

$\quad$ `skip`

$\quad \{\ p\ \}$

`od`

$\{\ (\ \mathbb{P}(true) = 0\ )\ +\ p\ \}$

$\Rightarrow \{\ p\ \}$

$\Rightarrow \{\ Inv\ \}$

En p se satisface $x = T$ con probabilidad 1, por lo tanto cuando se corta al predicado por la condición $(x < T)$ se obtiene $\mathbb{P}(true) = 0$; ya que se esta recortando todo el estado probabilístico. La tripla

$$\{\ \mathbb{P}(true) = 0\ \}\ MillerRabin1;\ prime := prime \wedge \neg witness;\ x := x + 1\ \{\ \mathbb{P}(true) = 0\ \}$$

se satisface trivialmente porque ninguna variable modifica el predicado $\mathbb{P}(true) = 0$.

Por otra parte cuando se corta al predicado p por la condición $\neg(x < T)$, no se altera porque $x = T$ implica $\neg(x < T)$. Notar que en la rama negativa del `if` se aplicó la regla (Skip).

Usando el lema 13 se obtiene que $(\ \mathbb{P}(true) = 0\ )\ +\ p$ es equivalente a p . La última implicación se deduce de las definiciones de Inv y p.

Capítulo 4

Lógica de Morgan

4.1 Extensión de la lógica de *Hoare*

Aquí se presentará el formalismo *pCGL*, desarrollado por Morgan y Seidel entre otros [5],[6],[7], [8]. Solo se hará referencias a programas determinísticos. Es decir que no se contemplarán aquellos programas que incluyan el operador de *no-determinismo* $\sqcap$.

Definición 14 *Supongamos que S ranguea sobre los programas, B sobre expresiones booleanas y p sobre expresiones rales en el rango $[0,1]$; asumamos que x representa una lista de variables distintas, y E una lista de expresiones (enteras o boolenas). Entonces la sintaxis del Lenguaje de Comandos Guardados Probabilísticos $PCGL$ esta dado por la siguiente gramática.*

$$S \ ::= \ \texttt{abort} \ | \ \texttt{skip} \ | \ x := E \ | \ S;S \ | \ \texttt{if} \ B \ \texttt{then} \ S \ \texttt{else} \ S \ \texttt{fi} \ | \ \texttt{do} \ B \to S \ \texttt{od} \ | \ S \oplus_p S$$

Todos los constructores, excepto el último, son convencionales [1]. El nuevo constructor introducido es el de selección probabilística: dado p en el intervalo cerrado $[0,1]$, escribimos $S \oplus_p T$ para la selección probabilística entre los programas S y T; los cuales tiene probabilidad de p y $1-p$ respectivamente de ser seleccionados. En la mayoría de los casos p será una contante ρ con $0 \leq \rho \leq 1$, pero puede ser más general; por ejemplo el programa $S \oplus_{[B]} T$ es equivalente a la selección condicional: $\texttt{if} \ B \ \texttt{then} \ S \ \texttt{else} \ T \ \texttt{fi}$. Notar que el constructor de iteración $\texttt{do}$, puede definirse como el menor punto fijo de una función:

$$\begin{array}{rcl} \texttt{do} \ B \to S \ \texttt{od} & := & \mu \ \mathbf{F}_{PCGL} \\ \text{donde:} & & \mathbf{F} : PCGL \to PCGL \\ & & \mathbf{F}.x \ := \ \texttt{if} \ B \ \texttt{then} \ (S;x) \ \texttt{else skip fi} \end{array}$$

Ejemplo 16 *Suponiendo que n tiene K bits, el siguiente programa retorna en la variable m un número con distribución uniforme en el rango $\left[0, 2^K\right)$.*

$$
\begin{array}{l}
\{\ \textit{suposición: } n = N\ \} \\
\left.\begin{array}{l}
k, m := n, 0; \\
\texttt{do}\ (k \neq 0) \longrightarrow \\
\qquad k := k\ \texttt{div}\ 2; \\
\qquad b := 0\ \oplus_{\frac{1}{2}}\ b := 1; \\
\qquad m := 2m + b \\
\texttt{od}
\end{array}\right\} UnifPow2_n
\end{array}
$$

•

Antes de introducir la probabilidad, se repasará la semántica del tranformador de estados wp para programas estandard (que no involucran el operador $\oplus_p$). Supongamos que St es el conjunto de estados sobre el cual el programa opera. Los predicados estandard son funciones del conjunto de estados S al conjunto de booleanos $\mathbb{B} = \{True, False\}$ y están parcialmente ordenados por la implicación.

$$\mathsf{P}(St) \ :=\ \big(St \to \mathbb{B}, \Rightarrow\ \big)$$

Un programa estandard es un *trasformador de predicados*, una función que mapea una *postcondición* (un predicado sobre estados finales) a la *precondición más débil* (un predicado sobre estados inciales): la precondición más débil identifica todos los estados inciales desde los cuales se garantiza que el programa termine satisfaciendo la postcondición. Luego la semántica del transformador de predicados estandard, está dada por el espacio de la función $\mathcal{D}(St)$ definida:

$$\mathcal{D}(St) \ :=\ \big(\mathsf{P}(St) \to \mathsf{P}(St), \sqsubseteq\ \big)$$

el orden $\sqsubseteq$ es derivado punto a punto del orden sobre predicados. Dada una postcondición Q y un programa S, escribimos $wp.S.Q$ para la precondición más débil de S con respecto a Q.

$$
\begin{array}{rcl}
wp.\ \texttt{skip}\ .Q & := & Q \\
wp.(x := E).Q & := & Q[x := E] \\
wp.(S;T).Q & := & wp.S.(wp.T.Q) \\
wp.(\ \texttt{if}\ B\ \texttt{then}\ S\ \texttt{else}\ T\ \texttt{fi}\ ).Q & := & (B \wedge wp.S.Q)\ \vee\ (\neg B \wedge wp.T.Q) \\
wp.(\ \texttt{do}\ B \to S\ \texttt{od}\ ) & := & \mu_{\mathcal{D}(St)}\mathbf{F} \\
wp.\ \texttt{abort}\ .Q & := & false
\end{array}
$$

donde: $\mu\mathbf{F}_{\mathcal{D}(St)}$ denota el menor punto fijo de:

$$
\begin{array}{l}
\mathbf{F} : \mathcal{D}(St) \to \mathcal{D}(St) \\
\mathbf{F}.t.Q \ :=\ (B \wedge wp.(t.Q)) \vee (\neg B \wedge Q) \qquad \text{con:}\quad t : \mathcal{D}(St)\ ,\ Q : \mathsf{P}(St)
\end{array}
$$

o expresado de otra manera:

$$\mathbf{F}.(wp.T) \ :=\ wp.(\ \texttt{if}\ B\ \texttt{then}\ S; T\ \texttt{else skip fi}\ )$$

A fin de introducir probabilidad, generalizamos los predicados. Recordemos que los predicados estandard pueden ser equivalentemente definidos como funciones características $St \to \{0,1\}$ con el orden punto a punto del $\leq$. Usaremos corchetes [.] para indicar esa situación, tal que $[true]$ y $[false]$ son las funciones constantes 1 y 0 sobre las variables de programa. De esta manera, las operaciones lógicas estandard pueden ser trasladadas de varias maneras en las operaciones aritmeticas. Más adelante se establecerá de manera más precisa la equivalencia entre ellas.

Definición 15 *El espacio de predicados probabilísticos sobre el conjunto de estados St se define:*

$$\mathcal{P}(St) \;:=\; \big(St \to \mathbb{R}_{\geq}, \Rrightarrow\big)$$

donde $\mathbb{R}_{\geq}$ denota los reales no negativos, y la relación $\Rrightarrow$ heredada punto a punto del ordenamiento $\leq$ usual de $\mathbb{R}_{\geq}$. El modelo de transformadores probabilísticos de estados es:

$$\mathcal{T}(St) \;:=\; \big(\mathcal{P}(St) \to \mathcal{P}(St), \sqsubseteq\big)$$

donde el orden $\sqsubseteq$ es derivado punto a punto de la implicación $\Rrightarrow$ sobre $\mathcal{P}(St)$.

Ejemplo 17 *Supongamos que $s \in St$ es un estado definido como $s := \langle b = 0\rangle$; y que $Q \in \mathcal{P}(St)$ es un predicado probabilístico definido como $Q := [b=0] * \frac{1}{2} + [b=1] * \frac{1}{2}$. Entonces se tienen las siguientes relaciones:*

$$Q.s \quad \equiv \quad \Big([b=0] * \frac{1}{2} + [b=1] * \frac{1}{2} \Big).\langle b=0\rangle \quad \equiv \quad \frac{1}{2} \quad \Rrightarrow \quad [true]$$

•

En lugar de actuar sobre predicados estandard, los comandos *PCGL* actúan sobre predicados probabilísticos, los cuales toman valores en el rango $[0,1]$. A continuación se dará una semántica probabilística a los constructores del lenguaje *PCGL*.

Definición 16 *Sea $Q : \mathcal{P}(St)$ un predicado probabilístico, entonces la semántica de los constructores del lenguage* PCGL*, está dada por las siguientes definiciones:*

$$\begin{array}{rcll}
wp.\,\texttt{skip}\,.Q & := & Q & \textbf{\textit{D1}} \\
wp.(x := E).Q & := & Q[x := E] & \textbf{\textit{D2}} \\
wp.(S;T).Q & := & wp.S.(wp.T.Q) & \textbf{\textit{D3}} \\
wp.(S \oplus_p T).Q & := & p * wp.S.Q \;+\; (1-p) * wp.T.Q & \textbf{\textit{D4}} \\
wp.(\,\texttt{if}\; B \;\texttt{then}\; S \;\texttt{else}\; T \;\texttt{fi}\,).Q & := & [B] * wp.S.Q \;+\; [\neg B] * wp.T.Q & \textbf{\textit{D5}} \\
wp.(\,\texttt{do}\; B \to S \;\texttt{od}\,) & := & \mu_{\mathcal{T}(St)}\mathbf{F} & \textbf{\textit{D6}} \\
wp.\,\texttt{abort}\,.Q & := & 0 & \textbf{\textit{D7}}
\end{array}$$

donde: $\mu\mathbf{F}_{\mathcal{T}(St)}$ denota el menor punto fijo de:

$$\begin{array}{l}
\mathbf{F} : \mathcal{T}(St) \to \mathcal{T}(St) \\
\mathbf{F}.t.Q \;:=\; [B] * wp.(t.Q) + [\neg B] * Q \qquad \textit{con:}\quad t : \mathcal{T}(St)\;,\; Q : \mathcal{P}(St)
\end{array}$$

o expresado de otra manera:

$$\mathbf{F}.(wp.T) \;:=\; wp.(\,\texttt{if}\; B \;\texttt{then}\; S;T \;\texttt{else skip fi}\,)$$

Notar que: Q no es necesariamente de la forma $[P]$, con P predicado estandard; $*$ es la multiplicación usual (que puede reemplazarse por $\sqcap$); $+$ es la suma usual (que puede reemplazarse por $\sqcup$); p es una expresión sobre variables de programa (no necesariamente una constante), que toma su valor en el rango $[0,1]$; y x es una variable o un vector de variables.

El programa `abort` no puede garantizarse que termine en algún estado, por lo tanto mapea cada postcondición a 0. El programa que termina inmediatamente `skip` no cambia nada, por lo tanto la postcondición Q es válida luego de la ejecución de `skip` solo si valía anteriormente. La precondición de la asignación $x := E$ es $Q[x := E]$, es decir el reemplazo sintáctico de todas las ocurrencias libres de x en la expresión Q por E, renombrando las variables ligadas en E si es necesario para evitar la captura de variables libres en E. La composición secuencial es la composición funcional. La precondición de la selección probabilística es la suma de las precondiciones de ambas ramas multiplicadas por pesos apropiados. Finalmente la semántica del programa recursivo `while` está dada como el menor punto fijo de una función.

Ejemplo 18 *Sea I el predicado probabilístico definido como:*

$$I \;:=\; \left[m2^{log2.k} \leq i < (m+1)2^{log2.k}\right] * 2^{-log2.k}$$

entonces:

$$\begin{array}{l}
\quad wp.(k := k \texttt{ div } 2;\ b := 0 \oplus_{\frac{1}{2}} b := 1;\ m := 2m+b).I \\
\equiv \{\ \textit{definición } \textbf{\textit{D3}}\ ,\ \textit{2 veces}\ \} \\
\quad wp.(k := k \texttt{ div } 2).\ wp.(b := 0 \oplus_{\frac{1}{2}} b := 1).\ wp.(m := 2m+b).I \\
\equiv \{\ \textit{definición } \textbf{\textit{D2}}\ \} \\
\quad wp.(k := k \texttt{ div } 2).wp.(b := 0 \oplus_{\frac{1}{2}} b := 1). \\
\qquad [(2m+b)2^{log2.k} \leq i < (2m+b+1)2^{log2.k}] * 2^{-log2.k} \\
\equiv \{\ \textit{definición } \textbf{\textit{D4}}\ \} \\
\quad wp.(k := k \texttt{ div } 2).(\ \tfrac{1}{2} * [2m2^{log2.k} \leq i < (2m+1)2^{log2.k}] * 2^{-log2.k} \\
\qquad +\ \tfrac{1}{2} * [(2m+1)2^{log2.k} \leq i < (2m+1+1)2^{log2.k}] * 2^{-log2.k}\ ) \\
\equiv \{\ \textit{definición } \textbf{\textit{D2}}\ \} \\
\quad \tfrac{1}{2} * [2m2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)} \quad + \\
\quad \tfrac{1}{2} * [(2m+1)2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1+1)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)}
\end{array}$$

•

En [1] Dijkstra impone ciertas condiciones saludables sobre los transformadores de estados estandard en $\mathcal{D}(St)$, algunas de ellas son:

$wp.S.(Q \wedge Q') \;\equiv\; wp.S.Q \;\wedge\; wp.S.Q'$	**conjuntividad**
$wp.S.false \;\equiv\; false$	**factibilidad**
si $\quad Q \Rightarrow Q' \quad$ entonces $\quad wp.S.Q \;\Rightarrow\; wp.S.Q'$	**monotonia**

Las condiciones son importantes porque ellas caracterizan exactamente aquellos programas que pueden dar una semántica alternativa, como relaciones entre estados inciales y finales; también pueden ser usadas

para probar propiedades generales para razonar con los programas. A continuación se consideran las propiedades análogas para los programas $PCGL$; y además algunas propiedades que son útiles para la manipulación de operadores lógicos y aritméticos.

Definición 17 *Sean $Q, Q' \in \mathcal{P}(St)$ predicacados probabilísticos; y $s \in St$ un estado, entonces se definen los siguientes operadores binarios*

a) *El mínimo*	$(Q \sqcap Q').s$	$:=$	$Q.s$ `min` $Q'.s$
b) *El máximo*	$(Q \sqcup Q').s$	$:=$	$Q.s$ `max` $Q'.s$
c) *La resta truncada*	$Q \ominus Q'$	$:=$	$(Q - Q') \sqcup 0$
d) *La conjunción probabilística*	$Q \& Q'$	$:=$	$(Q + Q') \ominus 1$

Seguidamente se dará una lista de propiedades que caracterizan la lógica de *Morgan*. Sea S un programa $PCGL$; Q, Q' predicados probabilísticos; a, b, c reales no negativos; $P, \overline{P}, P'$ predicados estandard, donde $P, \overline{P}$ son disjuntos, es decir: $P \wedge \overline{P} = false$; entonces las siguientes propiedades son válidas:

$wp.S.(a * Q + b * Q' \ominus c) \Leftleftarrows a * wp.S.Q + b * wp.S.Q' \ominus c$	**P1**
$wp.S.(Q \& Q') \Leftleftarrows wp.S.Q \ \& \ wp.T.Q'$	**P2**
si $Q' \Leftleftarrows Q$ entonces $wp.S.Q' \Leftleftarrows wp.S.Q$	**P3**
si $0 \leq c \leq 1$ entonces $wp.S.(c * Q) \equiv c * wp.S.Q$	**P4**
$a * [P] + b * [\overline{P}] \equiv a * [P] \sqcup b * [\overline{P}]$	**P5**
$P \Rightarrow P'$ si y solo si $[P] \Rrightarrow [P']$	**P6**
$[P \wedge P'] \equiv [P] * [P']$	**P7**
$[P \wedge P'] \equiv [P] \sqcap [P']$	**P8**
$[P \vee P'] \equiv [P] \sqcup [P']$	**P9**
si S no modifica a Q entonces: $Q \equiv wp.S.Q$	**P10**

Notar que las propiedades **P5** , **P6** , **P7** , **P8** , **P9** son consecuencia de como las operaciones lógicas sobre predicados estandard son trasladadas a las operaciones aritméticas; todas son aplicadas punto a punto. La propiedad fundamental **P1** se denomina *sub-linearidad*; a partir de ella se pueden probar **P2** , **P3** y **P4** , estas pruebas serán justificadas en el apéndice *Lemas de Morgan*. Notar que en $PCGL$, las condiciones saludables de [1] se convierten en sub-linearidad [5], la cual es una generalización de la conjuntividad. La propiedad **P2** se denomina *sub-conjuntividad* y representa una conjunción probabilística. La propiedad **P3** es la versión probabilística de la *monotonía*. La propiedad **P4** se denomina *escalado*, y sirve justamente para escalar un predicado probabilístico por una constante c. La propiedad **P10** dice que Q permanece igual si el programa S no modifica las variables de de programa de Q. Notar que en esta última propiedad se asume que el programa termina con con probabilidad 1 ($wp.S.1 \equiv 1$); ya que por ejemplo: $0 \equiv wp.$ `abort` $.Q$.

Ejemplo 19 *Aquí se muestra como funcionan algunas de las propiedades enunciadas:*

$$\tfrac{1}{2} * [2m2^{log2.(k \ \mathtt{div}\ 2)} \leq i < (2m+1)2^{log2.(k \ \mathtt{div}\ 2)}] * 2^{-log2.(k \ \mathtt{div}\ 2)} \ + \\ \tfrac{1}{2} * [(2m+1)2^{log2.(k \ \mathtt{div}\ 2)} \leq i < (2m+1+1)2^{log2.(k \ \mathtt{div}\ 2)}] * 2^{-log2.(k \ \mathtt{div}\ 2)}$$

$\equiv$ { *algebra* }

$[2m2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)-1} \ +$

$[(2m+1)2^{log2.(k \texttt{ div } 2)} \leq i < (2m+2)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)-1}$

$\equiv$ { *propiedad* ***P5*** }

$[2m2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)-1} \ \sqcup$

$[(2m+1)2^{log2.(k \texttt{ div } 2)} \leq i < (2m+2)2^{log2.(k \texttt{ div } 2)}] * 2^{-log2.(k \texttt{ div } 2)-1}$

$\equiv$ { *propiedad de* $\sqcup$ }

$\big($ $[2m2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1)2^{log2.(k \texttt{ div } 2)}] \ \sqcup$

$[(2m+1)2^{log2.(k \texttt{ div } 2)} \leq i < (2m+2)2^{log2.(k \texttt{ div } 2)}]$ $\big) * 2^{-log2.(k \texttt{ div } 2)-1}$

$\equiv$ { *propiedad* ***P9*** }

$\big($ $[$ $(2m2^{log2.(k \texttt{ div } 2)} \leq i < (2m+1)2^{log2.(k \texttt{ div } 2)}) \ \vee$

$((2m+1)2^{log2.(k \texttt{ div } 2)} \leq i < (2m+2)2^{log2.(k \texttt{ div } 2)})$ $]$ $\big) * 2^{-log2.(k \texttt{ div } 2)-1}$

$\equiv$ { *partición de rango; algebra* }

$[$ $(m2^{log2.(k \texttt{ div } 2)+1} \leq i < (m+1)2^{log2.(k \texttt{ div } 2)+1})$ $] * 2^{-log2.(k \texttt{ div } 2)-1}$

$\Leftarrow$ { $k \neq 0$ *; definición de* $log2$ *; propiedad* ***P6*** }

$[$ $m2^{log2.k} \leq i < (m+1)2^{log2.k} \ \wedge \ k \neq 0$ $] * 2^{-log2.k}$

$\equiv$ { *propiedad* ***P8*** }

$\big($ $\lceil m2^{log2.k} \leq i < (m+1)2^{log2.k} \rceil \ \sqcap \ [k \neq 0]$ $\big) * 2^{-log2.k}$

$\equiv$ { $[k \neq 0]$ *es estandard* }

$\lceil m2^{log2.k} \leq i < (m+1)2^{log2.k} \rceil * 2^{-log2.k} \ \sqcap \ [k \neq 0]$

•

A continuación se presenta la noción de corrección probabilística para ciclos de la forma:

$$loop \ := \ \texttt{do} \ G \rightarrow body \ \texttt{od}$$

En [8] se demuestra una regla, la cual establece que para ciclos *determinísticos* es suficiente con calcular las cotas de cada iteracion i separadamente y luego sumar los resultados. Notar que en la semántica estandard se explotaba el determinismo usando la disjunción de los resultados, aquí se lo hace con la suma. En la verifiación de $Uniform_a[2, n-1]$ se verá una aplicación interesante de esta regla.

Regla 1 (para ciclos determinísticos) *Sea Q un predicado probabilítico y body un programa determinístico, entonces:*

$$wp.loop.Q \ \equiv \ \sum_{i=0}^{\infty} \mathbf{h}^i.([\neg G] * Q) \qquad \textit{donde:} \quad \mathbf{h}.Q' \ := \ [B] * wp.body.Q'$$

En [6] hay dos reglas útiles para probar propiedades sobre ciclos. La primera de ellas determina una cota inferior para la probabilidad de que el ciclo *loop* termine en cierto estado. La segunda regla muestra la terminación del ciclo, usando un variante (evaluado en las variables de programa); el cual está acotado inferior y superiormente tal que en cada iteración se garantiza un decrecimiento estricto con una mínima probabilidad fija $p \neq 0$. Notar que se permite que el variante probabilístico crezca, pero no mas allá de la cota superior H.

Regla 2 (para ciclos probabilísticos) *Sea T la probabilidad de que el ciclo loop termine, definida como:*

$$T \;:=\; wp.loop.1$$

entonces corrección parcial (preservación del invariante) implica corrección total, siempre cuando el invariante I no exceda la condición de terminación T; es decir:

$$\begin{array}{ll} Si & I \sqcap [G] \;\Rrightarrow\; wp.body.I \\ y & I \Rrightarrow T \\ entonces & I \;\Rrightarrow\; wp.loop.([\neg G] \sqcap I) \end{array}$$

Ejemplo 20 *La primer condición es la invarianza de I con respecto al cuerpo del ciclo:*

$wp.\big(\; k := k \;\texttt{div}\; 2;\;\; b := 0 \oplus_{\frac{1}{2}} b := 1;\;\; m := 2m + b \;\big).I$

$\equiv$ { *ejemplo 18* }

$\frac{1}{2} * [2m2^{log2.(k \;\texttt{div}\; 2)} \leq i < (2m+1)2^{log2.(k \;\texttt{div}\; 2)}] * 2^{-log2.(k \;\texttt{div}\; 2)} \;+$

$\frac{1}{2} * [(2m+1)2^{log2.(k \;\texttt{div}\; 2)} \leq i < (2m+1+1)2^{log2.(k \;\texttt{div}\; 2)}] * 2^{-log2.(k \;\texttt{div}\; 2)}$

$\Leftarrow$ { *ejemplo 19* }

$\big[m2^{log2.k} \leq i < (m+1)2^{log2.k}\big] * 2^{-log2.k} \;\sqcap\; [k \neq 0]$

$\equiv$ { *definición de I* }

$I \sqcap [k \neq 0]$

La segunda condición asegura corrección total y como se verá en el ejemplo 21: $T \equiv 1$; con lo cual trivialmente se cumple que: $I \Rrightarrow T$. Por lo tanto usando la regla para ciclos probabilísticos, se tiene:

$$I \;\Rrightarrow\; wp.loop.([\neg(k \neq 0)] \sqcap I) \qquad (1)$$

Este resultado nos permite calular una cota inferior para la probabilidad de que $UnifPow2_n$ genere un número aleatorio en el rango $\big[0, 2^{log2.N}\big)$. Puede chequearse que para la salida del ciclo vale:

$$[\neg(k \neq 0)] \sqcap I \quad \equiv \quad [m = i] \qquad (2)$$

Por lo tanto:

$wp.UnifPow2_n.[m = i]$

$\equiv$ { *resultado (2)* }

$wp.UnifPow2_n.([\neg(k \neq 0)] \sqcap I)$

$\equiv$ { *definición **D3*** }

$wp.(k, m := n, 0)\;.wp.loop.([\neg(k \neq 0)] \sqcap I)$

$\Leftarrow$ { *resultado (1) ; propiedad **P3*** }

$wp.(k, m := n, 0).I$

$\equiv$ { *definición **D2*** }

$[0 \leq i < 2^{log2.n}] * 2^{-log2.n}$

$\equiv$ { *suposición inicial: $n = N$* }

$[0 \leq i < 2^{log2.N}] * 2^{-log2.N}$

Lo cual se interpreta como que el programa $UnifPow2_n$ establece la postcondición:

$[m = i]$ *con probabilidad de al menos* $2^{-log2.N}$ *cuando el predicado* $[0 \leq i < 2^{log2.N}]$ *es verdadero; y con probabilidad* 0 *en caso contrario.*

•

Regla 3 (de variantes probabilísticos) *Sea* V *una expresión entera evaluada en las variables de programa, definida al menos sobre algún subconjunto* I *del espacio de estados* S*, que para el ciclo loop y además:*

1. *existen constantes enteras fijas* L*(low) y* H*(high) tales que:*

$$G \sqcap I \;\; \Rrightarrow \;\; [L \leq V \leq H]$$

2. *el subconjunto* I *(como predicado estandard) es* $wp-$*invariante para el ciclo loop.*

3. *para alguna probabilidad fija* $p \neq 0$ *y para todos los enteros* K *se tiene:*

$$p * (G \sqcap I \sqcap [V = K]) \;\; \Rrightarrow \;\; wp.body.[V < K]$$

Entonces la terminación del ciclo es efectiva desde algun estado en el cual se satisface I*: tenemos* $I \Rrightarrow T$*, donde* T *es la condición de terminación del ciclo loop.*

Ejemplo 21 *En este caso se aplicará la regla de variantes probabilísticos al programa* $UnifPow2_n$ *. Definiendo* $V, L, H, p := k, 0, N, 1$ *y proponiendo como invariante estandard* $I' := [0 \leq k \leq N]$ *se cumplen:*

1. *existen constantes enteras fijas* L, H *tales que:* $[k \neq 0] \sqcap I' \;\; \Rrightarrow \;\; [L \leq k \leq H]$

2. *el predicado estandard* I' *es* $wp-$*invariante:*

$wp.(k := k \texttt{ div } 2; b := 0 \oplus_{\frac{1}{2}} b := 1; m := 2 * m + b).I'$

$\equiv$ { *definición* ***D3*** *, 2 veces* }

$wp.(k := k \texttt{ div } 2).\ wp.(b := 0 \oplus_{\frac{1}{2}} b := 1).\ wp(m := 2 * m + b).I'$

$\equiv$ { *definición* ***D2*** }

$wp.(k := k \texttt{ div } 2).\ wp.(b := 0 \oplus_{\frac{1}{2}} b := 1).I'$

$\equiv$ { *definición* ***D4*** }

$wp.(k := k \texttt{ div } 2).\big(\ \frac{1}{2} * wp.(b := 0).I' \ + \ \frac{1}{2} * wp.(b := 1).I' \ \big)$

$\equiv$ { *definición* ***D2*** *, 2 veces* }

$wp.(k := k \texttt{ div } 2).\big(\ \frac{1}{2} * I' \ + \ \frac{1}{2} * I' \ \big)$

$\equiv$ { *cálculo de predicados probabilísticos* }

$wp.(k := k \texttt{ div } 2).I'$

$\equiv$ { *definición* ***D2*** }

$[0 \leq k \texttt{ div } 2 \leq N]$

$\Lleftarrow$ { *propiedad* ***P6*** }

$I' \sqcap [k \neq 0]$

3. *La probabilidad fija* $p = 1 \neq 0$ *satisface:*

$$
\begin{array}{ll}
 & wp.(k := k \texttt{ div } 2; b := 0 \oplus_{\frac{1}{2}} b := 1; m := 2 * m + b).[k < K] \\
\equiv & \{ \textit{ idem. a los 5 primeros pasos de la prueba anterior } \} \\
 & wp.(k := k \texttt{ div } 2).[k < K] \\
\equiv & \{ \textit{ definición } \textbf{\textit{D2}} \} \\
 & [k \texttt{ div } 2 < K] \\
\Leftarrow & \{ \textit{ algebra ; propiedad } \textbf{\textit{P6}} \} \\
 & [k \neq 0 \wedge k = K] \\
\equiv & \{ \textit{ propiedad } \textbf{\textit{P8}} \} \\
 & [k \neq 0] \sqcap [k = K] \\
\Leftarrow & \{ \textit{ propiedad de } \sqcap \} \\
 & 1 * ([k \neq 0] \sqcap I' \sqcap [k = K])
\end{array}
$$

por lo tanto se deduce que si se parte desde un estado que satisface I' *la terminación es efectiva. Ello lo asegura la inicialización del ciclo, junto a la suposición inicial* $n = N$*:*

$$wp.(k, m := n, 0).I' \quad \equiv \quad [0 \leq n \leq N] \quad \equiv \quad 1$$

Es decir que con probabildad 1 *el ciclo parte de un estado que satisface* I'*, por lo tanto* $T \equiv 1$*.*

•

4.2 Verficiación de *FactorTwos*

$$
\begin{array}{l}
\{ \textbf{S1} \ : \ n = N \ \} \\
\quad r, s := 0, n - 1; \\
\quad \texttt{do } (s \texttt{ mod } 2 = 0) \rightarrow \\
\qquad\quad s := s \texttt{ div } 2; \\
\qquad\quad r := r + 1 \\
\quad \texttt{od}
\end{array}
\Bigg\} \ loop
$$

Terminación del ciclo

La terminación del ciclo se prueba usando la regla de variante probabilístico tomando:

$$V, L, H, p \ := \ s, 0, N-1, 1$$

y tomando a $[true]$ como invariante estandard sobre el cual está definido V.

Invariante probabilístico del ciclo

$$I \ := \ [n - 1 = 2^r s]$$

Salida del ciclo

$I \sqcap [\neg(s \texttt{ mod } 2 = 0)]$

$\equiv$ { s impar }

$[n-1 = 2^r s] \sqcap [\texttt{odd}.s]$

$\equiv$ { propiedad **P8** }

$[n-1 = 2^r s \ \wedge \ \texttt{odd}.s]$

Cuerpo del ciclo

$wp.(s := s \texttt{ div } 2;\ r := r+1).I$

$\equiv$ { definición **D3** }

$wp.(s := s \texttt{ div } 2).wp.(r := r+1).I$

$\equiv$ { definición **D2** }

$wp.(s := s \texttt{ div } 2).[n-1 = 2^{r+1} s]$

$\equiv$ { algebra }

$wp.(s := s \texttt{ div } 2).[n-1 = 2^r 2s]$

$\equiv$ { definición **D2** }

$[n-1 = 2^r 2(s \texttt{ div } 2)]$

$\Leftarrow$ { $s = 2(s \texttt{ div } 2) \ \Leftarrow \texttt{odd}.s$; propiedad **P6** }

$[n-1 = 2^r s \wedge s \texttt{ mod } 2 = 0]$

$\equiv$ { propiedad **P8** }

$[n-1 = 2^r s] \sqcap [s \texttt{ mod } 2 = 0]$

Inicialización del ciclo

$wp.(r, s := 0, n-1).I$

$\equiv$ { definición **D2** }

$[n-1 = 2^0 (n-1)]$

$\equiv$ { algebra }

$[true]$

Usando la regla para ciclos probabilísticos se obtiene:

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

Lo cual se interpreta como que el programa $FactorTwos$ establece la postcondición $[n-1 = 2^r s \wedge \texttt{odd}.s]$ con probabilidad 1.

4.3 Verficiación de $Uniform_a[2, n-1]$

$$
\begin{array}{l}
\{\ \textbf{S1:}\ n = N\ \} \\
\quad m := n; \\
\quad \texttt{do}\ \neg(2 \leq m < n)\ \longrightarrow \\
\qquad UnifPow2_n \\
\quad \texttt{od}\ ; \\
\quad a := m
\end{array}
\Bigg\} loop
$$

Antes de chequear la terminación del ciclo se probará un lema auxiliar que establece propiedades sobre una función que resultará útil para definir un variante probabilístico del ciclo.

Lema 1 *Sea* $f_N : \mathbb{R} \to \mathbb{R}_{\geq}$ *definida por* $f_N(x) = \left(x - \frac{N+1}{2}\right)^2$ *entonces:*

a) $x \in [2, N) \quad \Rightarrow \quad f_N(x) < \left(\frac{N-1}{2}\right)^2$

b) $x \notin [2, N) \quad \Rightarrow \quad f_N(x) \geq \left(\frac{N-1}{2}\right)^2$

c) $x \notin [2, N)$ *y* $x \in \left[0, 2^{log2.N}\right) \quad \Rightarrow \quad \left(\frac{N-1}{2}\right)^2 \leq f_N(x) \leq \left(2^{log2.N} - \frac{N-3}{2}\right)^2$

Dem:

Analizando la derivada de la función se puede ver en que intervalos la función es creciente (decreciente).

$$
\begin{array}{lllll}
x > \frac{N+1}{2} & \Rightarrow & f'_N(x) = 2x - (N+1) > 0 & \Rightarrow & f_N \ \text{es creciente en} \quad \left(\frac{N+1}{2}, +\infty\right) \quad (1) \\
x < \frac{N+1}{2} & \Rightarrow & f'_N(x) = 2x - (N+1) < 0 & \Rightarrow & f_N \ \text{es decreciente en} \quad \left(-\infty, \frac{N+1}{2}\right) \quad (2)
\end{array}
$$

Usando (1) y (2) se prueban a), b) y c).

a)

$$
\begin{array}{lll}
x \in [2, N) \ \Rightarrow & & f_N(x) \\
& < & f_N(1)\ \texttt{max}\ f_N(N) \\
& = & \left(1 - \frac{N+1}{2}\right)^2\ \texttt{max}\ \left(N - \frac{N+1}{2}\right)^2 \\
& = & \left(\frac{N-1}{2}\right)^2 \quad \bullet
\end{array}
$$

b)

$$
\begin{array}{lll}
x \notin [2, N) \ \Rightarrow & & f_N(x) \\
& \geq & f_N(1)\ \texttt{max}\ f_N(N) \\
& = & \left(1 - \frac{N+1}{2}\right)^2\ \texttt{max}\ \left(N - \frac{N+1}{2}\right)^2 \\
& = & \left(\frac{N-1}{2}\right)^2 \quad \bullet
\end{array}
$$

c) La primer desigualdad es un caso particular de B), mientras que:

$$
\begin{array}{lll}
x \notin [2, N) \text{ y } x \in \left[0, 2^{log2.N}\right) \ \Rightarrow & & f_N(x) \\
& \leq & f_N(0)\ \texttt{max}\ f_N\left(2^{log2.N} - 1\right) \\
& = & \left(0 - \frac{N+1}{2}\right)^2\ \texttt{max}\ \left(2^{log2.N} - 1 - \frac{N+1}{2}\right)^2 \\
& = & \left(2^{log2.N} - \frac{N-3}{2}\right)^2 \quad \bullet
\end{array}
$$

■

El siguiente reesultado se usará con frecuencia: la mínima probabilidad de que la variable m sea igual a alguno de los valores del conjunto de enteros I, luego de ejecutar $UnifPow2_n$ es de $2^{-log2.N}$ multiplicado por la cantidad de valores del conjunto I que están el rango $\big[0, 2^{log2.N}\big)$.

Lema 2 *Sea $I \subseteq \mathbb{Z}$ un subconjunto de enteros finito, entonces:*

$$2^{-log2.N} * \sum_{i \in I} [0 \leq i < 2^{log2.N}] \quad \Rrightarrow \quad wp.UnifPow2_n.\Big[\bigvee_{i \in I} (m = i)\Big]$$

Dem:

$wp.UnifPow2_n.\Big[\bigvee_{i \in I}(m = i)\Big]$
$\equiv$ { propiedad **P9** }
$wp.UnifPow2_n.\Big(\bigsqcup_{i \in I}[m = i]\Big)$
$\equiv$ { propiedad **P5** }
$wp.UnifPow2_n.\Big(\sum_{i \in I}[m = i]\Big)$
$\Lleftarrow$ { propiedad **P1** }
$\sum_{i \in I} wp.UnifPow2_n.[m = i]$
$\Lleftarrow$ { resultado anterior }
$\sum_{i \in I}[0 \leq i < 2^{log2.N}] * 2^{-log2.N}$
$\equiv$ { algebra }
$2^{-log2.N} * \sum_{i \in I}[0 \leq i < 2^{log2.N}]$

■

Terminación del ciclo

Para verificar la terminación del ciclo usamos la regla de variantes; para ello se definen:

$$\begin{array}{lcllcl} V & := & \big(m - \frac{N+1}{2}\big)^2 & p & := & (N-2)2^{-log2.N} \\ L & := & \big(\frac{N-1}{2}\big)^2 & I & := & \big[0 \leq m < 2^{log2.N}\big] \\ H & := & \big(2^{log2.N} - \frac{N-3}{2}\big)^2 & & & \end{array}$$

1. Existen constantes enteras L, H tales que: $\quad G \sqcap I \quad \Rrightarrow \quad [L < V < H]$

   $G \sqcap I$
   $\equiv$ { definición de G, I }
   $[\neg(2 \leq m < n)\ ] \sqcap \big[0 \leq m < 2^{log2.N}\big]$
   $\equiv$ { propiedad **P8** }
   $\big[\neg(2 \leq m < n) \ \wedge\ \big(0 \leq m < 2^{log2.N}\big)\ \big]$
   $\Rrightarrow$ { $n = N$, lema 1.c) }
   $\big[\ \big(\frac{N-1}{2}\big)^2 < \big(m - \frac{N+1}{2}\big)^2 < \big(2^{log2.N} - \frac{N-3}{2}\big)^2\ \big]$
   $\equiv$ { definición de L, V, H }
   $[L < V < H]$ •

2. El predicado estandard I es wp-invariante para el ciclo $loop$.

$wp.UnifPow2_n.I$

$\equiv$ { cálculo de predicados estandard }

$wp.UnifPow2_n.\Big[\bigvee_{i\in J}(m=i)\Big]$ donde $J := \{i \in \mathbb{Z} \ : \ 0 \leq i < 2^{log2.N}\}$

$\Leftarrow$ { lema 2 }

$2^{-log2.N} * \sum_{i\in J}[0 \leq i < 2^{log2.N}]$

$\equiv$ { $\#J = 2^{log2.N}$ }

$2^{-log2.N} * 2^{log2.N}$

$\Leftarrow$ { $1 \Leftarrow Q$ }

$[\neg(2 \leq m < n)\,] \sqcap I$ •

3. Para alguna probabilidad fija $p \neq 0$ y para todo entero K se tiene:

$wp.body.[V < K]$

$\equiv$ { definiciones de $body$ y V }

$wp.UnifPow2_n.[\ (m - \frac{N+1}{2})^2 < K\ ]$

$\Leftarrow$ { lema 1.a ; propiedad **P6** }

$wp.UnifPow2_n.[\ (2 \leq m < n) \wedge K > (\frac{N-1}{2})^2\ ]$

$\equiv$ { propiedad **P8** }

$wp.UnifPow2_n.(\ [2 \leq m < n] \sqcap [\ K \geq (\frac{N-1}{2})^2\ ]\ )$

$\Leftarrow$ { propiedad **P2** }

$wp.UnifPow2_n.[2 \leq m < n] \ \sqcap \ wp.UnifPow2_n.[\ K \geq (\frac{N-1}{2})^2\ ]$

$\equiv$ { propiedad **P10** }

$wp.UnifPow2_n.[2 \leq m < n] \ \sqcap \ [\ K \geq (\frac{N-1}{2})^2\ ]$

$\equiv$ { $n = N$; cálculo de predicados estandard }

$wp.UnifPow2_n.\Big[\bigvee_{i\in J}(m=i)\Big] \ \sqcap \ [\ K \geq (\frac{N-1}{2})^2\ ]$

donde $J := \{i \in \mathbb{Z} \ : \ 2 \leq i < N\}$

$\Leftarrow$ { lema 2 }

$2^{-log2.N} * \sum_{i\in J}[0 \leq i < 2^{log2.N}] \ \sqcap \ [\ K \geq (\frac{N-1}{2})^2\ ]$

$\equiv$ { $\#J = N-2$ }

$(N-2) * 2^{-log2.N} \sqcap [\ K \geq (\frac{N-1}{2})^2\ ]$

$\Leftarrow$ { propiedad de $\sqcap$ }

$(N-2) * 2^{-log2.N} * [\ K \geq (\frac{N-1}{2})^2\ ]$

$\Leftarrow$ { lema 1.b ; propiedad **P6** }

$(N-2) * 2^{-log2.N} * [\neg(2 \leq m < n) \wedge 0 \leq m < 2^{log2.N} \wedge (m - \frac{N+1}{2})^2 = K\ ]$

$\equiv$ { propiedad **P8** }

$(N-2) * 2^{-log2.N} * (\ [\neg(2 \leq m < n)] \sqcap [0 \leq m < 2^{log2.N}] \sqcap [\ (m - \frac{N+1}{2})^2 = K\ ]\ )$

$\equiv$ { definiciones de p, G, I, V }

$p * \big(G \sqcap I \sqcap [V = K]\big)$ •

Por lo tanto se satisfacen las tres condiciones de la regla de variantes probabilísticos, con lo cual se

obtiene que $I \Rrightarrow T$. Luego:

$$
\begin{array}{l}
\quad wp.Uniform_a[2, n-1].1 \\
\equiv \{ \text{ definición de } Uniform_a[2, n-1] \ \} \\
\quad wp.(m := n; loop; a := m).1 \\
\equiv \{ \text{ definición } \mathbf{D3} \ \} \\
\quad wp.(m := n).\ wp.loop.\ wp.(a := m).1 \\
\equiv \{ \text{ definición } \mathbf{D2} \ \} \\
\quad wp.(m := n).\ wp.loop.1 \\
\Leftarrow \{ \text{propiedad } \mathbf{P3} \ ; \ T \Leftarrow I \ \} \\
\quad wp.(m := n).[0 \leq m < 2^{log2.N}] \\
\equiv \{ \text{ definición } \mathbf{D2} \ \} \\
\quad [0 \leq n < 2^{log2.N}] \\
\equiv \{ \ n = N \ ; \text{ propiedad de } log2 \ \} \\
\quad 1
\end{array}
$$

Estableciendo de esta manera que el programa $wp.Uniform_a[2, n-1]$ termina con probabilidad 1.

Cálculo de $wp.Uniform_a[2, n-1].[a = i]$

A continuación calcularemos una cota inferior para la probabilidad de que el programa $Uniform_a[2, n-1]$ establezca la postcondición $[a = i]$, con $i = 2, 3, \ldots, N-1$. En el siguiente lema se calcula la función **h** definida en la regla para ciclos determinísticos.

Lema 3 *Sea i tal que $2 \leq i < N$ entonces para toda iteración j, $j \geq 1$, se tiene que:*

$$\mathbf{h}^j\big([2 \leq m < n] * [m = i]\big) \quad \Leftarrow \quad \frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\ ]$$

Dem:

La prueba será por inducción en j, tal que j es el número de iteración.

- **caso** $j = 1$

$$
\begin{array}{l}
\mathbf{h}^1\big([2 \leq m < n] * [m = i]\big) \\
\equiv \{ \text{ definición de } \mathbf{h} \ \} \\
\quad [\neg(2 \leq m < n)\ ] * wp.UnifPow2_n.\big([2 \leq m < n] * [m = i]\big) \\
\equiv \{ \text{ propiedad } \mathbf{P7} \ \} \\
\quad [\neg(2 \leq m < n)\ ] * wp.UnifPow2_n.[2 \leq m < n \wedge m = i] \\
\equiv \{ \ n = N \ \} \\
\quad [\neg(2 \leq m < n)\ ] * wp.UnifPow2_n.[2 \leq i < N \wedge m = i] \\
\equiv \{ \text{ propiedad } \mathbf{P8} \ \} \\
\quad [\neg(2 \leq m < n)\ ] * wp.UnifPow2_n.\big([2 \leq i < N] \sqcap [m = i]\big) \\
\equiv \{ \text{ propiedad } \mathbf{P2} \ \} \\
\quad [\neg(2 \leq m < n)\ ] * \big(\ wp.UnifPow2_n.[2 \leq i < N] \ \sqcap \ wp.UnifPow2_n.[m = i]\ \big) \\
\equiv \{ \text{ propiedad } \mathbf{P10} \ \}
\end{array}
$$

$[\neg(2 \leq m < n)\,] * (\ [2 \leq i < N]\ \sqcap\ wp.UnifPow2_n.[m = i]\ )$

$\Leftarrow$ { resultado anterior }

$[\neg(2 \leq m < n)\,] * (\ [2 \leq i < N]\ \sqcap\ [2 \leq i < N] * 2^{-log2.N}\ )$

$\equiv$ { propiedad de $\sqcap$; definición de p }

$[\neg(2 \leq m < n)\,] * [2 \leq i < N] * \frac{p}{N-2}$

$\equiv$ { algebra }

$\frac{p(1-p)^{1-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N]$ $\quad\bullet$

- **caso** $j+1$

$\mathbf{h}^{j+1}\big([2 \leq m < n] * [m = i]\big)$

$\equiv$ { definición de $\mathbf{h}$ }

$[\neg(2 \leq m < n)\,] * wp.UnifPow2_n.(\ \mathbf{h}^{j}\big([2 \leq m < n] * [m = i]\big)\ )$

$\Leftarrow$ { hipótesis inductiva }

$[\neg(2 \leq m < n)\,] * wp.UnifPow2_n.(\ \frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)] * [2 \leq i < N]\ )$

$\equiv$ { propiedad **P4** }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * wp.UnifPow2_n.(\ [\neg(2 \leq m < n)] * [2 \leq i < N]\ )$

$\equiv$ { propiedades **P7, P8** }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * wp.UnifPow2_n.(\ [\neg(2 \leq m < n)] \sqcap [2 \leq i < N]\ )$

$\equiv$ { propiedad **P2** }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * (\ wp.UnifPow2_n.[\neg(2 \leq m < n)]\ \sqcap\ wp.UnifPow2_n.[2 \leq i < N]\ )$

$\equiv$ { propiedad **P10** }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * (\ wp.UnifPow2_n.[\neg(2 \leq m < n)]\ \sqcap\ [2 \leq i < N]\ )$

$\Leftarrow$ { cálculo de predicados probabilísticos }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N] * wp.UnifPow2_n.[\neg(2 \leq m < n)]$

$\Leftarrow$ { $n = N$; propiedad **P6** }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N] * wp.UnifPow2_n.\Big[\ \bigvee_{i \in K}(m = i)\ \Big]$

donde: $\quad K := \{\ i \in \mathbb{Z}\ :\ 0 \leq i < 2 \quad \vee \quad N \leq i < 2^{log2.N}\ \}$

$\Leftarrow$ { lema 2 }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N] * 2^{-log2.N} * \sum_{i \in K}[0 \leq i < 2^{log2.N}]$

$\equiv$ { cálculo de #K }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N] * 2^{-log2.N} * (\ 2^{log2.N} - (N-2)\ )$

$\equiv$ { algebra }

$\frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N] * (\ 1 - (N-2) * 2^{-log2.N}\ )$

$\equiv$ { definición de p ; algebra }

$\frac{p(1-p)^{j}}{N-2} * [\neg(2 \leq m < n)\,] * [2 \leq i < N]$ $\quad\bullet$

■

Ahora si estamos en condiciones de dar una cota inferior para la probabilidad de que el programa $Uniform_a[2, n-1]$ establezca la postcondición $[a = i]$ con $i = 2, 3, \ldots, N-1$.

$wp.Uniform_a[2, n-1].[a = i]$

$\equiv$ { definición de $Uniform_a[2, n-1]$ }

$wp.(m := n; loop; a := m).[a = i]$

$\equiv$ { definición **D3** }

$wp.(m := n).\ wp.loop.\ wp.(a := m).[a = i]$

$\equiv$ { definición **D2** }

$wp.(m := n).\ wp.loop.[m = i]$

$\equiv$ { regla para ciclos determinísticos }

$wp.(m := n).(\ \sum_{j=0}^{\infty} \mathbf{h}^j\,([2 \le m < n] * [m = i])\ )$

$\Leftarrow$ { propiedad **P3** ; lema anterior }

$wp.(m := n).(\ [2 \le m < n] * [m = i]\ +\ \sum_{j=1}^{\infty} \frac{p(1-p)^{j-1}}{N-2} * [\neg(2 \le m < n)] * [2 \le i < N]\ )$

$\equiv$ { algebra }

$wp.(m := n).(\ [2 \le m < n] * [m = i]\ +\ \frac{p}{N-2} * [\neg(2 \le m < n)] * [2 \le i < N] * \sum_{j=0}^{\infty}(1-p)^j\ )$

$\equiv$ { $(1-p) \neq 0$; serie geométrica }

$wp.(m := n).(\ [2 \le m < n] * [m = i]\ +\ \frac{p}{N-2} * [\neg(2 \le m < n)] * [2 \le i < N] * \frac{1}{1-(1-p)}\ )$

$\equiv$ { algebra }

$wp.(m := n).(\ [2 \le m < n] * [m = i]\ +\ [\neg(2 \le m < n)] * [2 \le i < N] * \frac{1}{N-2}\ )$

$\equiv$ { definición **D2** }

$[2 \le n < n] * [n = i]\ +\ [\neg(2 \le n < n)] * [2 \le i < N] * \frac{1}{N-2}$

$\equiv$ { cálculo de predicados estandard }

$[false] * [n = i] + [\neg(false)] * [2 \le i < N] * \frac{1}{N-2}$

$\equiv$ { $[false] = 0$; $[true] = 1$}

$[2 \le i < N] * \frac{1}{N-2}$

A continuación se prueba un resultado similar al que se probó para el programa $UnifPow2_n$.

Lema 4 *Sea $I \subseteq \mathbb{Z}$ un subconjunto de enteros finito, entonces:*

$$\frac{1}{N-2} * \sum_{i \in I} [2 \le i < N] \quad \Rrightarrow \quad wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I} (a = i)\Big]$$

Dem:

$wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i)\Big]$

$\equiv$ { propiedad **P9** }

$wp.Uniform_a[2, n-1].\Big(\bigsqcup_{i \in I}[a = i]\Big)$

$\equiv$ { propiedad **P5** }

$wp.Uniform_a[2, n-1].\Big(\sum_{i \in I}[a = i]\Big)$

$\Leftarrow$ { propiedad **P1** }

$\sum_{i \in I} wp.Uniform_a[2, n-1].[a = i]$

$\Leftarrow$ { resultado anterior }

$\sum_{i \in I} [2 \le i < N] * \frac{1}{N-2}$

$\equiv$ { algebra }

$\frac{1}{N-2} * \sum_{i \in I}[2 \leq i < N]$

■

Notar que de este lema se desprende otra forma de chequear que el programa $Uniform_a[2, n-1]$ termina con probabilidad 1. Tomando $I := \{2, \ldots, N-1\}$.

$$
\begin{array}{l}
\quad wp.Uniform_a[2, n-1].1 \\
\Leftarrow \{ \text{ propiedad } \mathbf{P6} \ \} \\
\quad wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i) \Big] \\
\Leftarrow \{ \text{ lema } \mathbf{??} \ \} \\
\quad \frac{1}{N-2} * \sum_{i \in I}[2 \leq i < N] \\
\equiv \{ \ \#I = N-2 \ \} \\
\quad \frac{1}{N-2} * (N-2) \\
\equiv \{ \text{ algebra } \} \\
\quad 1 \qquad \bullet
\end{array}
$$

Por lo tanto tenemos que:

$$[2 \leq i < N] * \tfrac{1}{N-2} \ \Rrightarrow \ wp.Uniform_a[2, n-1].[a = i]$$

lo cual se interpreta como que el programa $Uniform_a[2, n-1]$ almacena en la variable a la constante i con probabilidad de al menos $\frac{1}{N-2}$ cuando i está en el rango $[2, N)$; y 0 en caso contrario.

4.4 Verficiación de *WitnessTail*

$$
\begin{array}{l}
\{ \ \mathbf{S1} \ : \quad n = N \ \wedge \ N \geq 3 \ \wedge \ \mathtt{odd}.N \ \wedge \ 2^r s = n-1 \ \wedge \ \mathtt{odd}.s \ \} \\
\quad \left.\begin{array}{l} a', y, witness := a^s \ \mathtt{mod}\ n, 0, false; \end{array}\right\} init \\
\quad \left.\begin{array}{l}
\mathtt{do}\ (y < r \wedge \neg witness) \ \longrightarrow \\
\qquad \mathtt{if}\ (a' \neq 1 \wedge a' \neq n-1)\ \mathtt{then} \\
\qquad\quad \left.\begin{array}{l} a' := a'^2 \ \mathtt{mod}\ n; \\ witness := (a' = 1) \end{array}\right\} S \\
\qquad \mathtt{else} \\
\qquad\quad \left.\begin{array}{l} a' := a'^2 \ \mathtt{mod}\ n \end{array}\right\} T \\
\qquad \mathtt{fi}\ ; \\
\qquad y := y + 1 \\
\mathtt{od}\ ;
\end{array}\right\} loop \\
\quad witness := witness \vee (a' \neq 1)
\end{array}
$$

Terminación del ciclo

La terminación del ciclo se prueba usando la regla de variante probabilístico tomando:

$$V, L, H, p \ := \ r-y, 0, N, 1$$

y tomando a $[0 \leq y \leq r]$ como invariante estandard sobre el cual está definido V. Usando la suposición **S1** se tiene que vale la desigualdad $1 < 2^r s < N$, la cual implica que $r < N$. Esta observación junto con el invariante estandard anterior asegura que: $0 \leq r - y \leq N$; justificandose así la elección de L, H. La asignación $y := y + 1$ asegura el decremento de V.

Invariante probabilístico del ciclo

Definimos los siguientes predicados, donde I es el invariante para $WitnessTail$.

$$\begin{array}{rcl} I & := & [y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))] \\ W(e) & := & \big(\exists j : 0 \leq j < e : a^{2^{j+1} s} \bmod n = 1 \wedge a^{2^j s} \bmod n \neq 1 \wedge a^{2^j s} \bmod n \neq n-1\big) \end{array}$$

Notar que de la definición de $W(y)$ se deduce que:

$$\begin{array}{rcl} W(0) & \equiv & false \\ W(y+1) & \equiv & W(y) \vee (a^{2^{y+1} s} \bmod n = 1 \wedge a^{2^y s} \bmod n \neq 1 \wedge a^{2^y s} \bmod n \neq n-1) \\ \mathsf{witness}.n.a & \equiv & W(r) \vee a^{2^r s} \bmod n \neq 1 \quad \equiv \quad W(r) \vee a^{n-1} \bmod n \neq 1 \end{array}$$

Salida del ciclo

$[\neg(y < r \wedge \neg witness)] \sqcap I$

$\equiv$ { cálculo de predicados ; propiedad **P8** }

$[(y \geq r \vee witness) \ \wedge \ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))]$

$\equiv$ { distributividad de $\vee$ }

$[\ (y \geq r \ \wedge y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ) \ \vee$

$(witness \ \wedge y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ )\ ]$

$\equiv$ { propiedad **P9** }

$[\ y \geq r \ \wedge y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ] \ \sqcup$

$[\ witness \ \wedge y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ]$

$\Rrightarrow$ { $y = r$, $2^r s = n-1$, en el termino izquierdo }

$[\ a' = a^{n-1} \bmod n \wedge (witness \Leftrightarrow W(r))\ ] \ \sqcup$

$[\ witness \ \wedge y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ]$

$\Rrightarrow$ { $(A \wedge (A \Leftrightarrow B)) \quad \Rightarrow \quad ((A \vee A') \Leftrightarrow (B \vee B'))$; propiedad **P6** }

$[\ a' = a^{n-1} \bmod n \wedge (witness \Leftrightarrow W(r))\ ] \ \sqcup$

$[\ witness \vee (a' \neq 1) \Leftrightarrow W(r) \vee (a^{n-1} \bmod n \neq 1)\ ]$

$\Rrightarrow$ { $(A \Leftrightarrow B) \quad \Rightarrow \quad ((A \vee C) \Leftrightarrow (B \vee C))$; propiedad **P6** }

$[witness \vee (a' \neq 1) \Leftrightarrow W(r) \vee (a^{n-1} \bmod n \neq 1)] \ \sqcup$

$[witness \vee (a' \neq 1) \Leftrightarrow W(r) \vee (a^{n-1} \bmod n \neq 1)]$

$\equiv$ { propiedad de $\sqcup$ }

$[witness \vee (a' \neq 1) \Leftrightarrow W(r) \vee a^{n-1} \bmod n \neq 1]$

$\equiv$ { definición de $\mathsf{witness}.n.a$ }

$[witness \vee (a' \neq 1) \Leftrightarrow \mathsf{witness}.n.a]$

Cuerpo del ciclo

$wp.(\texttt{if } (a' \neq 1 \wedge a' \neq n-1) \texttt{ then } S \texttt{ else } T \texttt{ fi } ;\ y := y+1).$
$\quad [\ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ]$
$\equiv$ { definición **D3** }
$wp.(\texttt{if } (a' \neq 1 \wedge a' \neq n-1) \texttt{ then } S \texttt{ else } T \texttt{ fi }).wp(y := y+1).$
$\quad [\ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y))\ ]$
$\equiv$ { definición **D2** }
$wp.(\texttt{if } (a' \neq 1 \wedge a' \neq n-1) \texttt{ then } S \texttt{ else } T \texttt{ fi }).$
$\quad [\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { definición **D5** }
$(a' \neq 1 \wedge a' \neq n-1) * wp.S.[\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]\ +$
$\neg(a' \neq 1 \wedge a' \neq n-1) * wp.T.[\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { definiciones de S, T, definición **D3** }
$(a' \neq 1 \wedge a' \neq n-1) * wp.(a' := a'^2 \bmod n).wp.(witness := (a' = 1)).$
$\quad [\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]\ +$
$\neg(a' \neq 1 \wedge a' \neq n-1) * wp.(a' := a'^2 \bmod n).$
$\quad [\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { definición **D2** en ambos terminos }
$(a' \neq 1 \wedge a' \neq n-1) * wp.(a' := a'^2 \bmod n).$
$\quad [\ y+1 \leq r \wedge a' = a^{2^{y+1} s} \bmod n \wedge (a' = 1 \Leftrightarrow W(y+1))\ ]\ +$
$\neg(a' \neq 1 \wedge a' \neq n-1) *$
$\quad [\ y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { definición **D2** en el termino izquierdo }
$(a' \neq 1 \wedge a' \neq n-1) *$
$\quad [\ y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (a'^2 \bmod n = 1 \Leftrightarrow W(y+1))\ ]\ +$
$\neg(a' \neq 1 \wedge a' \neq n-1) *$
$\quad [\ y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { propiedad **P5** }
$[\ (a' \neq 1 \wedge a' \neq n-1)\ \wedge$
$\quad y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (a'^2 \bmod n = 1 \Leftrightarrow W(y+1))\ ]\ \sqcup$
$[\neg(a' \neq 1 \wedge a' \neq n-1)\ \wedge$
$\quad y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ ]$
$\equiv$ { propiedad **P9** }
$[\ (\ (a' \neq 1 \wedge a' \neq n-1)\ \wedge$
$\quad y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (a'^2 \bmod n = 1 \Leftrightarrow W(y+1))\ )\ \vee$
$(\neg(a' \neq 1 \wedge a' \neq n-1)\ \wedge$
$\quad y+1 \leq r \wedge a'^2 \bmod n = a^{2^{y+1} s} \bmod n \wedge (witness \Leftrightarrow W(y+1))\ )\ ]$
$\Leftarrow$ { Calculo de predicados ; Propiedad **P6** }
$[\ ((a' \neq 1 \wedge a' \neq n-1)\ \wedge$

$\neg witness \wedge y+1 \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y)) \ ) \ \vee$

$(\neg(a' \neq 1 \wedge a' \neq n-1) \ \wedge$

$\neg witness \wedge y+1 \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y)) \ ) \ ]$

$\equiv$ { calculo de predicados }

$[\ (y < r \wedge \neg witness) \ \wedge \ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y)) \ ]$

$\equiv$ { propiedad **P8** }

$[\ y < r \wedge \neg witness \ ] \sqcap [\ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y)) \ ]$

Inicialización del ciclo

$wp.init.I$

$\equiv$ { definiciones de $init, I$ }

$wp.(\ a', y, witness := a^s \bmod n, 0, false \ ). \ [\ y \leq r \wedge a' = a^{2^y s} \bmod n \wedge (witness \Leftrightarrow W(y)) \ ]$

$\equiv$ { definición **D2** }

$[\ 0 \leq r \wedge a^s \bmod n = a^{2^0 s} \bmod n \wedge (false \Leftrightarrow W(0)) \ ]$

$\equiv$ { suposición **S1** $\Rightarrow$ $0 \leq r$ }

$[true]$

Con lo cual obtenemos el siguiente resultado:

$$1 \ \Rrightarrow \ wp.(init; loop).[witness \vee (a' \neq 1) \Leftrightarrow \mathsf{witness}.n.a]$$

Finalmente:

$wp.WitnessTail.[witness \Leftrightarrow \mathsf{witness}.n.a]$

$\equiv$ { definición de $WitnessTail$ }

$wp.(init; loop; \ witness := witness \vee (a' \neq 1) \ ).[witness \Leftrightarrow \mathsf{witness}.n.a]$

$\equiv$ { definición **D3** }

$wp.(init; loop).wp.(witness := witness \vee (a' \neq 1)).[witness \Leftrightarrow \mathsf{witness}.n.a]$

$\equiv$ { definición **D2** }

$wp.(init; loop).[witness \vee (a' \neq 1) \Leftrightarrow \mathsf{witness}.n.a]$

$\Leftarrow$ { resultado anterior }

1

A continuación se probará un lema que resultará útil en la verificación posterior de $MillerRabin1$.

Lema 5 $\quad 1 \ \equiv \ wp.WitnessTail.[witness \Leftrightarrow \mathsf{witness}.n.a]$

implica que:

a) $[\ \mathsf{witness}.n.a] \ \equiv \ wp.WitnessTail.[\ witness]$

b) $[\neg \mathsf{witness}.n.a] \ \equiv \ wp.WitnessTail.[\neg witness]$

Dem:

Se probará el caso a); el caso b) es análogo.

$wp.WitnessTail.[witness]$
$\equiv$ { definición **D2** }
$wp.WitnessTail.[witness]$
$\Leftarrow$ { propiedad **P6** }
$wp.WitnessTail.[(witness \Leftrightarrow \mathsf{witness}.n.a) \wedge \mathsf{witness}.n.a]$
$\equiv$ { propiedad **P8** }
$wp.WitnessTail.([witness \Leftrightarrow \mathsf{witness}.n.a] \sqcap [\mathsf{witness}.n.a])$
$\Leftarrow$ { propiedad **P2** }
$wp.WitnessTail.[witness \Leftrightarrow \mathsf{witness}.n.a] \sqcap wp.WitnessTail.[\mathsf{witness}.n.a])$
$\equiv$ { hipótesis ; propiedad **P10** }
$(1 \sqcap [\mathsf{witness}.n.a])$
$\equiv$ { propiedad de $\sqcap$ }
$[\mathsf{witness}.n.a]$

■

Esto significa que el programa $WitnessTail$ setea la variable booleana $witness$ a $true$ con probabilidad 1 cuando la base a es un testigo de n; y 0 en caso contrario. Análogamente si a no es un testigo de n, con probabilidad 1 la variable $witness$ es $false$; y probabilidad 0 en caso contrario.

4.5 Verificación de $MillerRabin1$, con N primo

$$\{\ \mathbf{S1}\ :\quad n = N\ \wedge\ N \geq 3\ \wedge\ \mathsf{odd}.N\ \wedge\ \mathsf{prime}.N\ \wedge\ 2^r s = n-1\ \wedge\ \mathsf{odd}.s\ \}$$
$$\left.\begin{array}{l} Uniform_a[2, n-1]; \\ WitnessTail \end{array}\right\} MillerRabin1$$

Notar que la suposición **S1** es necesaria paraestablecer la suposición **S2** .

$wp.MillerRabin1.[\neg witness]$
$\equiv$ { definición de MillerRabin1 }
$wp.(Uniform_a[2, n-1]; WitnessTail).[\neg \mathsf{witness}.n.a]$
$\equiv$ { definición **D3** }
$wp.Uniform_a[2, n-1].\ wp.WitnessTail.[\neg \mathsf{witness}.n.a]$
$\equiv$ { lema 5.b }
$wp.Uniform_a[2, n-1].[\neg \mathsf{witness}.n.a]$
$\Leftarrow$ { $n = N$; propiedad **P6** }
$wp.Uniform_a[2, n-1].\left[\bigvee_{i \in I}(a = i \wedge \neg \mathsf{witness}.N.i)\right]$ donde: $I := \{2, \ldots, N-1\}$
$\equiv$ { suposición **S1** : $\mathsf{prime}.N \quad \Rightarrow \quad (\forall\ i : 0 < i < N :\ \neg \mathsf{witness}.N.i\ )$ }
$wp.Uniform_a[2, n-1].\left[\bigvee_{i \in I}(a = i)\right]$
$\Leftarrow$ { lema [?] }
$\frac{1}{N-2} * \sum_{i \in I}[2 \leq i < N]$

$\equiv$ { $\#I = N-2$ }
$\frac{1}{N-2} * (N-2)$
$\equiv$ { algebra }
1

Por lo tanto si N es primo, el programa $MillerRabin1$ setea la variable booleana $witness$ a $false$ con probabilidad 1:

$$1 \;\equiv\; wp.MillerRabin1.[\neg witness]$$

4.6 Verificación de $MillerRabin1$, con N compuesto

Recordemos la siguiente propiedad enunciada al comienzo:

$$N \geq 3 \wedge \mathsf{odd}.N \wedge \neg \mathsf{prime}.N \quad \Rightarrow \quad |\, \{i \;:\; 0 < i < N \wedge \mathsf{witness}.N.i \;\}\, | \geq \tfrac{N-1}{2}$$

Ella expresa que bajo ciertas condiciones de N, al menos $\frac{N-1}{2}$ bases son testigos. Ello nos permite hacer las siguientes suposiciones:

$\{$ **S1** : $n = N \;\wedge\; N \geq 3 \;\wedge\; \mathsf{odd}.N \;\wedge\; \neg\mathsf{prime}.N \;\wedge\; 2^r s = n-1 \;\wedge\; \mathsf{odd}.s \;\}$
$\{$ **S2** : $(\; 0 < A_1 < \ldots < A_K < N \;) \;\wedge\; (\; 0 < A_{K+1} < \ldots < A_{N-1} < N \;) \;\wedge\; K \geq \frac{N-1}{2} \;\wedge$
$\big(\; \forall i : 0 < i < N :\; \mathsf{witness}.N.i \Leftrightarrow i \in \{A_1, \ldots, A_K\} \;\wedge\; \neg\mathsf{witness}.N.i \Leftrightarrow i \in \{A_{K+1}, \ldots, A_{N-1}\} \;\big) \;\}$

$$\left.\begin{array}{l} Uniform_a[2, n-1]; \\ WitnessTail \end{array}\right\} MillerRabin1$$

Notar que la conjunción $(0 < A_1 < \ldots < A_K < N)$ expresa que en la secuencia $A_1, \cdots, A_K$ no hay constantes repetidas. Por otra parte, el predicado $witness.N.i \Leftrightarrow i \in \{A_1, \ldots, A_K\}$ expresa que si una constante en el rango $(0, N)$ es atestiguada entonces pertenece al conjunto $\{A_1, \ldots, A_K\}$ y viceversa. Recordemos que 1 siempre es una base no atestiguada para todo N. Por otra parte sabemos que existe un numero que denota la cantidad exacta de bases atestiguadas, lo denotaremos K y aunque no lo conocemos solo nos basta saber que es fijo y que es $\geq \frac{N-1}{2}$.

Se puede observar que:

- $1 \in \{A_{K+1}, \ldots, A_{N-1}\}$
- $\{A_1, \ldots, A_K\} \uplus \{A_{K+1}, \ldots, A_{N-1}\} = \{1, \ldots, N-1\}$
- $(A_1, \ldots, A_K, A_{K+1}, \ldots, A_{N-1})$ es alguna permutación de $(1, \ldots, N-1)$, donde las primeras K constantes son los testigos y las ultimas $N-1-K$ constantes no lo son.

A continuación se calcula la mínima probabilidad que el programa $MillerRabin1$ setee la variable booleana $witness$ a $true$:

$wp.MillerRabin1.[witness]$
$\equiv$ { definición de $MillerRabin1$ }

$wp.(Uniform_a[2, n-1]; WitnessTail).[witness]$

$\equiv$ { definición **D3** }

$wp.Uniform_a[2, n-1].\ wp.WitnessTail.[\neg \mathsf{witness}.n.a]$

$\equiv$ { lema 5.a }

$wp.Uniform_a[2, n-1].[\mathsf{witness}.n.a]$

$\Leftarrow$ { $n = N$; propiedad **P6** }

$wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i \wedge \mathsf{witness}.N.i)\Big]$ donde: $I := \{A_1, \ldots, A_K\}$

$\equiv$ { suposición **S2** }

$wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i)\Big]$

$\Leftarrow$ { lema **??** }

$\frac{1}{N-2} * \sum_{i \in I}[2 \leq i < N]$

$\equiv$ { $\#I = K$; algebra }

$\frac{K}{N-2}$

$\equiv$ { suposición **S2** : $K \geq \frac{N-1}{2}$; definimos $\epsilon := \frac{K}{N-2} - \frac{1}{2}$ }

$\frac{1}{2} + \epsilon$ $\quad \bullet$

Análogamente se tiene una prueba similar para el caso $\neg witness$:

$wp.MillerRabin1.[\neg witness]$

$\equiv$ { definición de MillerRabin1 }

$wp.(Uniform_a[2, n-1]; WitnessTail).[\neg \mathsf{witness}.n.a]$

$\equiv$ { definición **D3** }

$wp.Uniform_a[2, n-1].\ wp.WitnessTail.[\neg \mathsf{witness}.n.a]$

$\equiv$ { lema 5.b }

$wp.Uniform_a[2, n-1].[\neg \mathsf{witness}.n.a]$

$\Leftarrow$ { $n = N$; propiedad **P6** }

$wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i \wedge \neg \mathsf{witness}.N.i)\Big]$ donde: $I' := \{A_{K+1}, \ldots, A_{N-1}\}$

$\equiv$ { suposición **S2** }

$wp.Uniform_a[2, n-1].\Big[\bigvee_{i \in I}(a = i)\Big]$

$\Leftarrow$ { lema **??** }

$\frac{1}{N-2} * \sum_{i \in I'}[2 \leq i < N]$

$\equiv$ { $\#I' = N-1-K$ y $1 \in I'$ }

$\frac{1}{N-2} * (N-2-K)$

$\equiv$ { algebra }

$\frac{1}{2} - \left(\frac{K}{N-2} - \frac{1}{2}\right)$

$\equiv$ { definición anterior de ϵ }

$\frac{1}{2} - \epsilon$ $\quad \bullet$

Notar que:

- $\frac{K}{N-2} \geq (\frac{N-1}{2})(\frac{1}{N-2}) = \frac{N-1}{2(N-2)} > \frac{1}{2}$

- $0 < \epsilon < \frac{1}{2}$
- $0 < \frac{1}{2} - \epsilon < \frac{1}{2} < \frac{1}{2} + \epsilon < 1$

Por lo tanto cuando N es compuesto, el programa $MillerRabin1$ setea $witness$ a $false$ con probabilidad de al menos $(\frac{1}{2} - \epsilon)$; y setea $witness$ a $true$ con probabilidad de al menos $(\frac{1}{2} + \epsilon)$:

$$(\tfrac{1}{2} + \epsilon) \;\Rrightarrow\; wp.MillerRabin1.[\, witness]$$
$$(\tfrac{1}{2} - \epsilon) \;\Rrightarrow\; wp.MillerRabin1.[\neg witness]$$

4.7 Verificación de $MillerRabin$, con N primo

$\{$ **S1** : $\quad n = N \;\wedge\; N \geq 3 \;\wedge\; \mathsf{odd}.N \;\wedge\; \mathsf{prime}.N \;\wedge\; 2^r s = n - 1 \;\wedge\; \mathsf{odd}.s \;\}$
$x, prime := 0, true;$
`do` $(x < T) \;\longrightarrow$
    $MillerRabin1;$
    $prime := prime \wedge \neg witness;$
    $x := x + 1$
`od`

Terminación del ciclo

La terminación del ciclo se prueba usando la regla de variante probabilístico tomando:

$$V, L, H, p \;:=\; T - x, 0, T, 1$$

y tomando a $[0 \leq x \leq T]$ como invariante estandard sobre el cual está definido V. Dicho invariante implica que $0 \leq T - x \leq T$; y la asignación $x := x + 1$ asegura el decremento de V.

Invariante probabilístico del ciclo

$$I \;:=\; [x \leq T \wedge prime]$$

Salida del ciclo

$[x \leq T \wedge prime] \sqcap [\neg(x < T)]$
$\equiv$ { propiedad **P8** }
$[x \leq T \wedge prime \wedge x \geq T]$
$\equiv$ { cálculo de predicados }
$[x = T \wedge prime]$
$\Rrightarrow$ { propiedad **P6** }
$[prime]$

Cuerpo del ciclo

$$
\begin{array}{ll}
 & wp.(MillerRabin1;\ prime := prime \wedge \neg witness;\ x := x+1).I \\
\equiv & \{ \text{ definición } \mathbf{D3} \ \} \\
 & wp.MillerRabin1.wp.(prime := prime \wedge \neg witness).wp.(x := x+1).I \\
\equiv & \{ \text{ definición } \mathbf{D2} \ \} \\
 & wp.MillerRabin1.wp.(prime := prime \wedge \neg witness).[x+1 \leq T \wedge prime] \\
\equiv & \{ \text{ definición } \mathbf{D2} \ \} \\
 & wp.MillerRabin1.[x+1 \leq T \wedge prime \wedge \neg witness] \\
\equiv & \{ \text{ propiedad } \mathbf{P8} \ \} \\
 & wp.MillerRabin1.([x+1 \leq T \wedge prime] \sqcap [\neg witness]) \\
\Leftarrow & \{ \text{ propiedad } \mathbf{P2} \ \} \\
 & wp.MillerRabin1.[x+1 \leq T \wedge prime] \ \sqcap \ wp.MillerRabin1.[\neg witness] \\
\equiv & \{ \text{ propiedad } \mathbf{P10} \ \} \\
 & [x+1 \leq T \wedge prime] \ \sqcap \ wp.MillerRabin1.[\neg witness] \\
\equiv & \{ \text{ suposición } \ \mathbf{S1} \ \ ; \text{ verificación de } MillerRabin1 \ \} \\
 & [x+1 \leq T \wedge prime] \sqcap 1 \\
\equiv & \{ \text{ propiedad de } \sqcap \ \} \\
 & [x+1 \leq T \wedge prime] \\
\Leftarrow & \{ \text{ propiedad } \mathbf{P6} \ \} \\
 & [x \leq T \wedge prime \wedge x < T] \\
\equiv & \{ \text{ propiedad } \mathbf{P8} \ \} \\
 & [x \leq T \wedge prime] \sqcap [x < T]
\end{array}
$$

Inicialización del ciclo

$$
\begin{array}{ll}
 & wp.(\ x, prime := 0, true\ ).I \\
\equiv & \{ \text{ definición } \mathbf{D2} \ \} \\
 & [x \leq T] \\
\equiv & \{ \ 0 < T \ \} \\
 & [true]
\end{array}
$$

Por lo tanto el programa $MillerRabin$ establece con probabilidad 1 la postcondición $[prime]$ cuando asuminos como hipótesis global que la variable n almacena una contante N prima:

$$[\mathsf{prime}.n] \ \equiv \ wp.MillerRabin.[prime]$$

4.8 Verificación de $MillerRabin$, con N compuesto

$\{$ **S1** : $n = N \ \wedge \ N \geq 3 \ \wedge \ \mathsf{odd}.N \ \wedge \ \neg\mathsf{prime}.N \ \wedge \ 2^r s = n-1 \ \wedge \ \mathsf{odd}.s \ \}$
$x, prime := 0, true;$
do $(x < T) \ \longrightarrow$
  $MillerRabin1;$
  $prime := prime \wedge \neg witness;$
  $x := x + 1$
od

Terminación del ciclo

Idénticamente a la verificación de $MillerRabin$ para el caso N primo.

Invariante probabilístico del ciclo

$$I \ := \ [\neg prime] + [prime] * \left(1 - \tfrac{1}{2}^{T-x}\right)$$

Salida del ciclo

$[\neg(x < T)] \sqcap I$
$\Rrightarrow$ { cálculo de predicados estandard }
$[x = T] \ \sqcap \ \left(\ [\neg prime] + [prime] * \left(1 - \tfrac{1}{2}^{T-x}\right) \ \right)$
$\Rrightarrow$ { $x = T$; algebra }
$[\neg prime]$

Cuerpo del ciclo

$wp.(MillerRabin1; \ prime := prime \wedge \neg witness; \ x := x+1).I$
$\equiv$ { definición **D3** , dos veces}
$wp.MillerRabin1.wp.(prime := prime \wedge \neg witness).wp.(x := x+1).I$
$\equiv$ { definición **D2** }
$wp.MillerRabin1.wp.(prime := prime \wedge \neg witness).\left(\ [\neg prime] + [prime] * \left(1 - \tfrac{1}{2}^{T-x-1}\right) \ \right)$
$\equiv$ { definición **D2** }
$wp.MillerRabin1.\left(\ [\neg prime \vee witness] + [prime \wedge \neg witness] * \left(1 - \tfrac{1}{2}^{T-x-1}\right) \ \right)$
$\Lleftarrow$ { propiedad **P1** }
$wp.MillerRabin1.[\neg prime \vee witness] \ +$
$wp.MillerRabin1.\left(\ [prime \wedge \neg witness] * \left(1 - \tfrac{1}{2}^{T-x-1}\right) \ \right)$
$\Lleftarrow$ { propiedades **P9,P8** }
$wp.MillerRabin1.([\neg prime] \sqcup [witness]) \ +$
$wp.MillerRabin1.\left(\ ([prime] \sqcap [\neg witness]) * \left(1 - \tfrac{1}{2}^{T-x-1}\right) \ \right)$

$\Leftarrow$ { propiedad **P3** }

$\quad \big(\ wp.MillerRabin1.[\neg prime] \ \sqcup\ wp.MillerRabin1.[witness]\ \big)\ +$

$\quad wp.MillerRabin1.\big(\ ([prime] \sqcap [\neg witness]) * \big(1 - \frac{1}{2}^{T-x-1}\big)\ \big)$

$\Leftarrow$ { propiedad **P4** }

$\quad \big(\ wp.MillerRabin1.[\neg prime] \ \sqcup\ wp.MillerRabin1.[witness]\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * wp.MillerRabin1.([prime] \sqcap [\neg witness])$

$\Leftarrow$ { propiedad **P2** }

$\quad \big(\ wp.MillerRabin1.[\neg prime] \ \sqcup\ wp.MillerRabin1.[witness]\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * \big(\ wp.MillerRabin1.[prime] \ \sqcap\ wp.MillerRabin1.[\neg witness]\ \big)$

$\Leftarrow$ { propiedad **P10** }

$\quad \big(\ [\neg prime] \sqcup wp.MillerRabin1.[witness]\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * \big(\ [prime] \sqcap wp.MillerRabin1.[\neg witness]\ \big)$

$\Leftarrow$ { resultados anteriores }

$\quad \big(\ [\neg prime] \sqcup (\frac{1}{2} + \epsilon)\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * \big(\ [prime] \sqcap (\frac{1}{2} - \epsilon)\ \big)$

$\Leftarrow$ { propiedad **P4** }

$\quad \big(\ [\neg prime] \sqcup (\frac{1}{2} + \epsilon) * [prime]\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * \big(\ [prime] \sqcap (\frac{1}{2} - \epsilon) * [prime]\ \big)$

$\equiv$ { propiedad de $\sqcap$ }

$\quad \big(\ [\neg prime] \sqcup (\frac{1}{2} + \epsilon) * [prime]\ \big)\ +$

$\quad \big(1 - \frac{1}{2}^{T-x-1}\big) * (\frac{1}{2} - \epsilon) * [prime]$

$\Leftarrow$ { algebra }

$\quad \big(\ [\neg prime] \sqcup (\frac{1}{2} + \epsilon) * [prime]\ \big)\ +$

$\quad \big(\frac{1}{2} - \epsilon - \frac{1}{2}^{T-x}\big) * [prime]$

$\Leftarrow$ { propiedad **P5** }

$\quad [\neg prime] + (\frac{1}{2} + \epsilon) * [prime] +$

$\quad \big(\frac{1}{2} - \epsilon - \frac{1}{2}^{T-x}\big) * [prime]$

$\equiv$ { cálculo de predicados probabilísticos }

$\quad [\neg prime] + \big(1 - \frac{1}{2}^{T-x}\big) * [prime]$

$\Leftarrow$ { propiedad de $\sqcap$ }

$\quad \big(\ [\neg prime] + [prime] * \big(1 - \frac{1}{2}^{T-x}\big)\ \big)\ \sqcap\ [x < T]$

Inicialización del ciclo

$\quad wp.(x, prime := 0, true).I$

$\equiv$ { definición **D2** }

$\quad [\neg true] + [true] * \big(1 - \frac{1}{2}^{T-0}\big)$

$\equiv$ { $[false] = 0$; $[true] = 1$; algebra }

$\quad 1 - \frac{1}{2}^{T}$

Por lo tanto la mínima probabilidad de que el programa $MillerRabin$ setee la variable boleana $prime$ a $false$, es de $1 - \frac{1}{2}^T$ bajo la hipótesis global que la variable n almacena una constante N compuesta:

$$[\neg \mathsf{prime}.n] * \left(1 - 2^{-T}\right) \quad \Rrightarrow \quad wp.MillerRabin.[\neg prime]$$

Capítulo 5

Comparación de las lógicas

Este trabajo es un estudio comparativo en el cual se aplicó dos métodos formales distintos al mismo problema: analizar las propiedades probabilísticas que ya se sabía de ante mano que satisfacía el algoritmo de Miller-Rabin. Algunos de los objetivos de Hurd en [2], eran verificar el algortimo utilizando el probador de teoremas *HOL* y asegurar eficiencia trabajando sobre una versión en que la terminación es efectiva. El principal objetivo aquí no es la verificación en sí misma, sino comparar ambas lógicas y establecer las ventajas de cada una para razonar sobre programas probabilísticos. Además de ello se trabajó deliberadamente con una versión de Miller-Rabin que tiene terminación probabilística para estudiar como se comporta cada lógica ante esta situación. Notar que si la terminación es efectiva o probabilística depende de como esté definido el generador de números aleatorios, ya que es el único fragmento del programa que involucra la selección probabilística $\oplus_\rho$. Como resultado de las dos verificaciones se vió la importancia de utilizar alguna herramienta formal cuando se introduce probabilidad en los algoritmos, ya que resultados que parecen trivialmente ciertos pueden resultar falsos [1]. Los parámetros que se tuvieron en cuenta para compararar las dos lógicas son los detallados a continuación. Recordar que *pH* es el sistema de prueba creado por Hartog [4] y *pCGL* es la lógica desarrolada por Morgan ([5],[6], [7]) y Seidel [8] entre otros.

Expresividad de predicados Como dice Hartog en [4], la principal diferencia entre ambas lógicas es que en la suya se toma como punto de partida una lógica al estilo *Hoare* con predicados probabilísticos. Desde el punto de vista de la corrección estos predicados parecen intuitivamente más atractivos. El cálculo basado en *weakest preconditions* de Morgan provee una forma de generar mucha información útil, aplicando los tranformadores de predicados probabilísticos *wp.S*. Por otra parte el enfoque de *Hartog* hace énfasis en hacer la verificación a partir de mucha información, la cual es provista por la gran expresividad de sus predicados probabilísticos. Justamente proponer un invariante de ciclo en *pH* suele ser más sencillo que hacerlo en *pCGL*, ya que en la primera se puede representar la distribución de probabilidad de cada variable.

Tamaño de pruebas La extensión de *pCGL* se ha comportado mejor en este sentido. Las pruebas suelen ser más mecánicas y muchas veces resultan de sucesivas aplicaciones de los tranformadores

[1] En la lógica de *Hartog*, $q \oplus_\rho q$ no necesariamente implica q

de predicados probabilísticos $wp.S$. En la mayoría de las verificaciones realizadas aquí las pruebas terminan siendo más compactas.

En cambio en el sistema pH en programas de mayor tamaño el arbol de prueba se vuelve poco manejable, obligando a recurrir a pruebas *outline* y a documentar subpruebas en otra parte. En el apéndice de *hartog* puede observarse que cualquier prueba que se quiera hacer sobre predicados probabilísticos, hay que reducir el razonamiento a la tradicional lógica de primer orden; lo cual involucra muchos pasos de reducción.

Corrección total Para obtener corrección total es necesario verificar que el programa termine utilizando algún argumento probabilístico. Aquí se analizarán las condiciones de terminación para un ciclo probabilístico de la forma `while` C `do` S `od` . Recordemos que en todo este trabajo solo consideramos programas determinísticos; por lo tanto en la lógica $pCGL$ tenemos dos maneras de chequear la terminación de un ciclo:

1) Usar la regla para ciclos determinísticos y calcular directamente la condición de terminación:
$wp.($ `while` C `do` S `od` $).1 \quad \equiv \quad \sum_{i=0}^{\infty} \mathbf{h}^i.[\neg G]$

2) Usar la regla de variantes probabilísticos, utilizando algún invariante estandard I del ciclo y concluir que: $wp.($ `while` C `do` S `od` $).1 \quad \Leftarrow \quad I$. Con lo cual se obtiene que el ciclo termina con probabilidad 0 o 1 debido a que I es estandard.

En el sistema pH se fortalece la noción de invariante imponiendo una condición extra denominada $\langle C, S\rangle$-closedness. La idea detras de ella es que una secuencia se obtiene por repetidas iteraciones del ciclo, tendrá un supremo satisfaciendo el invariante. Cuando el ciclo termina en K iteraciones, un invariante que satisface corrección parcial automáticamente satisface $\langle C, S\rangle$-closedness (ver verificación de $UniformPow2_n$). De las verificaciones realizadas aquí solo hubo que chequear tal condición para el programa $Uniform_a[0, n-1]$, porque es el único que tiene una terminación probabilística.

La conclusión para este punto particular es que hay muchos pasos involucrados en el chequeo de la condición $\langle C, S\rangle$-closedness; los cuales no son tan mecánicos como en el caso de $pCGL$. En algún aspecto $pCGL$ parece ser mas flexible porque permite chequar la terminación de un algoritmo probabilístico de dos formas diferentes y mucho mas mecánias que el sistema pH.

Corrección Parcial Cuando se dice corrección parcial se hace referencia a la invarianza de algún predicado probabilístico con respecto al cuerpo del ciclo. Supongamos que I un predicado probabilístico en $pCGL$ y que Inv un predicado probabilístico en el sistema pH. En general resulta menos costoso demostrar que I es wp-invariante, que demostrar la tripla $\{\ Inv\ \}$ `if` C `then` S `else skip fi` $\{\ Inv\ \}$. Para el primer caso solo basta con probar: $[C] \sqcap I \ \Rrightarrow\ wp.S.I$. Para el segundo caso es necesario constuir el siguiente árbol de prueba:

$$\dfrac{\dfrac{\{\ C\ ?\ Inv\ \}\ S\ \{\ q\ \} \qquad \dfrac{\overline{\{\neg C\ ?\ Inv\ \}\ \texttt{skip}\ \{\neg C\ ?\ Inv\ \}}^{(Skip)} \quad (\neg C\ ?\ Inv)\ \Rightarrow\ q'}{\{\neg C\ ?\ Inv\ \}\ \texttt{skip}\ \{\ q'\ \}}^{(Cons)}}{\{\ Inv\ \}\ \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else skip fi}\ \{\ q + q'\ \}}^{(If)} \quad q + q'\ \Rightarrow\ Inv}{\{\ Inv\ \}\ \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else skip fi}\ \{\ Inv\ \}}^{(Cons)}$$

Mirando las hojas del árbol se conluye que se deben probar la tripla $\{\ C\ ?\ Inv\ \}\ S\ \{\ q\ \}$ y las implicaciones $(\neg C\ ?\ Inv) \Rightarrow q'$, $q + q' \Rightarrow Inv$. lo cual resulta mucho más costoso que en el caso de $pCGL$. Recordar que probar cualquier implicación entre dos predicados probabilísticos de pH obliga a reducir todo el razonamiento a la lógica de primer orden. Por otra parte no suele ser intuitivo ver que predicado probabilístico es impliacado por $(\neg C\ ?\ Inv)$.

Forma de acotar la probabilidad Este punto está relacionado de alguna manera con la expresividad de las lógicas; y también aquí se demuestra el poder de pH. Para fijar ideas supongamos que: $Pre, Post$ son predicados estandard en $pCGL$, es decir $Pre, Post \in \mathsf{P}(S)$; $Pre, Post$ también son predicados determinísticos en pH, es decir $Pre, Post \in DPred$; ρ es una constante en el intervalo $(0, 1)$; y que S es un programa probabilístico. Entonces en general, las lógicas puede acotar la probabilidad de las siguientes maneras:

Tipo de cota	pH	*pCGL*
inferior estricta	-	$\{\ \mathbb{P}(Pre) = 1\ \}\ S\ \{\ \mathbb{P}(Post) < \rho\ \}$
inferior	$\rho * [Pre] \ \Rrightarrow\ wp.S.[Post]$	$\{\ \mathbb{P}(Pre) = 1\ \}\ S\ \{\ \mathbb{P}(Post) \leq \rho\ \}$
exacta	$\rho * [Pre] \ \equiv\ wp.S.[Post]$	$\{\ \mathbb{P}(Pre) = 1\ \}\ S\ \{\ \mathbb{P}(Post) = \rho\ \}$
superior	-	$\{\ \mathbb{P}(Pre) = 1\ \}\ S\ \{\ \mathbb{P}(Post) \geq \rho\ \}$
superior estricta	-	$\{\ \mathbb{P}(Pre) = 1\ \}\ S\ \{\ \mathbb{P}(Post) > \rho\ \}$

Recordar que en la verificación de $MillerRabin$ para el caso $\neg\mathsf{prime}.N$ con la lógica de *Morgan*, fue necesario utilizar dos resultados:

$$(\tfrac{1}{2} + \epsilon) \ \Rrightarrow\ wp.MillerRabin1.[\,witness]$$
$$(\tfrac{1}{2} - \epsilon) \ \Rrightarrow\ wp.MillerRabin1.[\neg witness]$$

para ello se debió realizar dos pruebas; porque ninguno se podía deducir a partir del otro [2]. Sin embargo en la verificación de $MillerRabin1$ usando el sistema pH, ante una situación similar ocurrió lo contrario. Se disponia de la tripla:

$$\{\ \mathbb{P}(I \wedge \neg\mathsf{prime}.N) = 1\ \}\ MillerRabin1\ \{\ \mathbb{P}(witness) \geq \tfrac{1}{2}\ \wedge\ \mathbb{P}(true) = 1\ \}$$

y con solo aplicar la regla (Cons) a la postcondición de la tripla anterior, se obtiene la siguiente tripla:

$$\{\ \mathbb{P}(I \wedge \neg\mathsf{prime}.N) = 1\ \}\ MillerRabin1\ \{\ \mathbb{P}(\neg witness) \leq \tfrac{1}{2}\ \wedge\ \mathbb{P}(true) = 1\ \}$$

Por lo tanto la lógica pH es más poderosa en el sentido que se maneja más información. En muchos casos se pueden establacer una cota para la probabilidad de un predicado determinístico y al mismo tiempo una cota para la negación del mismo.

Programas verificables El conjunto de programas probabilísticos que se pueden verificar en $pCGL$ es mayor que en pH. Esto se debe al tipo de expresión por cuyo valor se puede realizar la selección probabilística:

[2] Notar que no existe una propiedad de la forma: $wp.S.(1-Q) \ \equiv\ 1 - wp.S.Q$, solo basta con tomar $S :=$ `abort`

Lógica	Selección Probabilística	Tipo de expresión
pH	$S \oplus_\rho T$	$\rho \in (0,1)$ es constante.
$pCGL$	$S \oplus_p T$	p es una expresión sobre variables de programa (no necesariamente constante) tomando un valor en $[0,1]$.

El siguiente programa es un generador de números aleatorios en el rango $[0, N)$ que esta extraído de [6].

$$Uniform \left\{ \begin{array}{l} a, b := 0, N; \\ \texttt{while } (a+1 \neq b) \texttt{ do} \\ \qquad p := (a+b) \texttt{ div } 2; \\ \qquad a := p \ \oplus_{\frac{b-p}{b-a}} \ b := p \qquad \longleftarrow (*) \\ \texttt{od} \end{array} \right.$$

Notar que el fragmento de programa (*) no se puede representar en pH porque la expresión $\frac{b-p}{b-a}$ no es una constante en el rango $(0,1)$, sino que depende de los valores de las variables a, b, p .

Otro ejemplo similar es la implementación de la selección determinística $\texttt{if } B \texttt{ then } S \texttt{ else } T \texttt{ fi}$ usando la selección probabilística $S \oplus_{[B]} T$. En $pCGL$ la equivalencia se muestra punto a punto:

$$wp.(\ S \oplus_{[B]} T\ ).Q \quad \equiv \quad [B] * wpS.Q \ + \ [\neg B] * wp.T.Q \quad \equiv \quad wp.(\ \texttt{if } B \texttt{ then } S \texttt{ else } T \texttt{ fi}\ ).Q$$

Sin embargo el programa $S \oplus_{[B]} T$ no pertenece al lenguaje $\mathcal{L}_{pw}$ de pH, porque el valor de la expresión depende las variables involucradas en la condición booleana B.

En resumen tenemos las siguientes comparaciones:

Punto de comparación	pH	$pCGL$
expresividad	$\surd$	$\times$
tamaño de pruebas	$\times$	$\surd$
corrección total	$\times$	$\surd$
corrección parcial	$\times$	$\surd$
forma de acotar la probabilidad	$\surd$	$\times$
programas verificables	$\times$	$\surd$

Como balance final, afirmar que una lógica es mejor que la otra podría resultar una opinión subjetiva; porque depende mucho del algoritmo probabilístico que se esté verificando. Para el caso particular del algoritmo de Miller-Rabin la lógica $pCGL$ resultó ser mucho más práctica, una vez que se maduró bién como trabajan los trasformadores de estados $wp.S$; todo el trabajo de verificación se hace casi mecánicamente. Sin embargo para una persona que nunca trabajó con alguna de los dos lógicas, podría verse tentada a empezar con la de Hartog, puesto que el constructor $\mathbb{P}(\ldots)$ está muy relacionado al concepto intuitivo de probabilidad que tiene la mayoría. $\heartsuit$

Apéndice A

Propiedades de la lógica de Hartog

En este apéndice demostraremos algunos lemas y propiedades necesarios para verificar el algoritmo de $MillerRabin$ usando la lógica de *Hartog* [4].

El primer lema es bastante intuitivo, y expresa que si debilitamos un predicado determinístico dp a uno dp', entonces la probabilidad de que sea cierto podría crecer.

Lema 6 $(\forall \sigma \in S, I \in \mathcal{I} : (\sigma, I) \vDash dp \rightarrow dp') \quad \Rightarrow \quad (\theta, J) \vDash \mathbb{P}(dp) \leq \mathbb{P}(dp')$

Dem:

$$
\begin{array}{l}
\quad (\forall \sigma \in S, I \in \mathcal{I} : (\sigma, I) \vDash dp \rightarrow dp') \\
\Leftrightarrow (\sigma', J_{\mathcal{I}}) \vDash dp \rightarrow dp' \\
\Leftrightarrow (\sigma', J_{\mathcal{I}}) \vDash dp \quad \Rightarrow \quad (\sigma', J_{\mathcal{I}}) \vDash dp' \\
\Leftrightarrow V \subseteq V' \\
\qquad \text{donde:} \quad V = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp\} \quad \text{y} \quad V' = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp'\} \\
\Rightarrow \sum_V \theta(\sigma) \leq \sum_{V'} \theta(\sigma) \\
\Leftrightarrow \mathcal{V}_r(\mathbb{P}(dp))(\theta, J) \ \leq \ \mathcal{V}_r(\mathbb{P}(dp'))(\theta, J) \\
\Leftrightarrow \mathcal{B}_r(\mathbb{P}(dp) \leq \mathbb{P}(dp'))(\theta, J) \\
\Leftrightarrow (\theta, J) \vDash \mathbb{P}(dp) \leq \mathbb{P}(dp')
\end{array}
$$

■

Para mostrar que la equivalencia no es cierta, se observa que se podría tener:
$(\theta, J) \vDash \mathbb{P}(true) \leq \mathbb{P}(false)$ y sin embargo: $(\sigma, I) \vDash true \rightarrow false$ es falso.

Supongamos que en cierto estado θ un predicado determinístico dp vale con cierta probabilidad, que está acotada inferiormente (superiormente) por cierto valor r; y además c es una condición booleana de programa. Entonces sería natural pensar que podemos dividir ese estado probabilístico en dos partes: una donde se satisface $dp \wedge c$ y otra donde se satisface $dp \wedge \neg c$. Luego podemos decir que existen cotas inferiores (superiores) para las probabilidades de esos predicados determinísticos disjuntos; cuya suma es $= r$. Esta idea motiva el siguiente lema que será muy útil en la verificación de programas de la forma: `if` c `then` $\cdots$ `else` $\cdots$ `fi` .

Lema 7 *Si* $c \in BC\langle PVar\rangle$; $J(r) \in [0,1]$ y $\trianglelefteq \in \{\leq, =, \geq\}$ *entonces:*
$(\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r \quad \Leftrightarrow \quad (\theta, J) \vDash \exists r_1, r_2 : r_1 + r_2 = r \ \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2$

Dem:

$(\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r$
$\Leftrightarrow \mathcal{B}_r(\mathbb{P}(dp) \trianglelefteq r)(\theta, J)$
$\Leftrightarrow \mathcal{V}_r(\mathbb{P}(dp))(\theta, J) \ \trianglelefteq \ \mathcal{V}_r(r)(\theta, J)$
$\Leftrightarrow \sum_{\sigma \in V} \theta(\sigma) \trianglelefteq J(r)$
donde: $V = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp\}$
$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : x_1 + x_2 = J(r) \ \wedge \ \sum_{\sigma \in V_1} \theta(\sigma) \trianglelefteq x_1 \ \wedge \ \sum_{\sigma \in V_2} \theta(\sigma) \trianglelefteq x_2$
donde: $V_1 = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge c\}$ y $V_2 = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge \neg c\}$

Subprueba:

notar que: $V = V_1 \uplus V_2$ es unión disjunta.

• **caso** $J(r) = 0$ tomamos:

$x_1 = x_2 = 0$

• **caso** $\trianglelefteq \in \{\leq\}$ y $J(r) > 0$ tomamos:

$x_1 = \sum_{\sigma \in V_1} \theta(\sigma) + \frac{1}{2}\big(J(r) - \sum_{\sigma \in V} \theta(\sigma)\big)$
$x_2 = \sum_{\sigma \in V_2} \theta(\sigma) + \frac{1}{2}\big(J(r) - \sum_{\sigma \in V} \theta(\sigma)\big)$

• **caso** $\trianglelefteq \in \{=, \geq\}$ y $J(r) > 0$ tomamos:

$x_1 = J(r)\big(\sum_{\sigma \in V_1} \theta(\sigma)\big) / \big(\sum_{\sigma \in V} \theta(\sigma)\big)$
$x_2 = J(r)\big(\sum_{\sigma \in V_2} \theta(\sigma)\big) / \big(\sum_{\sigma \in V} \theta(\sigma)\big)$

en todos los casos se satisface: $x_1 + x_2 = J(r)$ •

$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : \ J[r_1/x_1][r_2/x_2](r_1) + J[r_1/x_1][r_2/x_2](r_2) = J(r) \ \wedge$
$\sum_{\sigma \in V_1} \theta(\sigma) \trianglelefteq J[r_1/x_1][r_2/x_2](r_1) \ \wedge \ \sum_{\sigma \in V_2} \theta(\sigma) \trianglelefteq J[r_1/x_1][r_2/x_2](r_2)$
$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : \ \mathcal{V}_r(r_1 + r_2)(\theta, J[r_1/x_1][r_2/x_2]) \ = \ \mathcal{V}_r(r)(\theta, J[r_1/x_1][r_2/x_2]) \ \wedge$
$\mathcal{V}_r(\mathbb{P}(dp \wedge c))(\theta, J[r_1/x_1][r_2/x_2]) \ \trianglelefteq \ \mathcal{V}_r(r_1)(\theta, J[r_1/x_1][r_2/x_2]) \ \wedge$
$\mathcal{V}_r(\mathbb{P}(dp \wedge \neg c))(\theta, J[r_1/x_1][r_2/x_2]) \ \trianglelefteq \ \mathcal{V}_r(r_2)(\theta, J[r_1/x_1][r_2/x_2])$
$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : \ \mathcal{B}_r(r_1 + r_2 = r)(\theta, J[r_1/x_1][r_2/x_2]) \wedge$
$\mathcal{B}_r(\mathbb{P}(dp \wedge c) \trianglelefteq r_1)(\theta, J[r_1/x_1][r_2/x_2]) \ \wedge \ \mathcal{B}_r(\mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2)(\theta, J[r_1/x_1][r_2/x_2])$
$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : \ \mathcal{B}_r(\ r_1 + r_2 = r \ \wedge \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2 \ )(\theta, J[r_1/x_1][r_2/x_2])$
$\Leftrightarrow \exists x_1, x_2 \in \mathbb{R} : (\theta, J[r_1/x_1][r_2/x_2]) \vDash \ r_1 + r_2 = r \ \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2$
$\Leftrightarrow (\theta, J) \vDash \exists r_1, r_2 : \ r_1 + r_2 = r \ \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2$

■

Supongamos que estamos verificando un `if` y que ya hemos demostrado las siguientes triplas:

$$\{\ c?p\ \}\ s\ \{\ r_1 + r_2 = r \wedge \mathbb{P}(dp) \trianglelefteq r_1\ \} \qquad \text{y} \qquad \{\neg c?p\ \}\ s'\ \{\ r_1 + r_2 = r \wedge \mathbb{P}(dp) \trianglelefteq r_2\ \}$$

para algún predicado probabilístico p; entonces utilizando la regla (If) tenemos que vale la tripla:

$$\{\ p\ \}\ \texttt{if}\ c\ \ \texttt{then}\ s\ \texttt{else}\ s'\ \texttt{fi}\ \{\ (r_1 + r_2 = r \wedge \mathbb{P}(dp) \trianglelefteq r_1)\ +\ (r_1 + r_2 = r \wedge \mathbb{P}(dp) \trianglelefteq r_2)\ \}$$

Con lo cual sería bueno tener un lema que simplifique la suma probabilística en la postcondición de la tripla al predicado: $\mathbb{P}(dp) \trianglelefteq r$.

Lema 8 *Si* $J(r) \in [0,1]$ *y* $\trianglerighteq \in \{<, \leq, =, \geq, >\}$ *entonces:*
$(\theta, J) \vDash\ r_1 + r_2 = r \wedge \big(\mathbb{P}(dp) \trianglelefteq r_1\ +\ \mathbb{P}(dp) \trianglelefteq r_1\big) \quad \Rightarrow \quad (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r$

Dem:

$(\theta, J) \vDash\ r_1 + r_2 = r \wedge \big(\mathbb{P}(dp) \trianglelefteq r_1\ +\ \mathbb{P}(dp) \trianglelefteq r_1\big)$
$\Leftrightarrow (\theta, J) \vDash\ r_1 + r_2 = r\ \wedge\ (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r_1\ +\ \mathbb{P}(dp) \trianglelefteq r_1$
$\Leftrightarrow J(r_1) + J(r_2) = J(r)\ \wedge$
$\exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta\ \wedge\ (\theta_1, J) \vDash \mathbb{P}(dp) \trianglelefteq r_1\ \wedge\ (\theta_2, J) \vDash \mathbb{P}(dp) \trianglelefteq r_2$
$\Leftrightarrow J(r_1) + J(r_2) = J(r)\ \wedge$
$\exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta\ \wedge\ \sum_{\sigma \in V} \theta_1(\sigma) \trianglelefteq J(r_1)\ \wedge\ \sum_{\sigma \in V} \theta_1(\sigma) \trianglelefteq J(r_2)$
donde: $V = \{\sigma' | (\sigma', J_{\mathcal{I}}) \vDash dp\}$
$\Leftrightarrow J(r_1) + J(r_2) = J(r)\ \wedge\ \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta\ \wedge$
$\sum_{\sigma \in V} \theta(\sigma) = \sum_{\sigma \in V} (\theta_1 + \theta_2)(\sigma) = \sum_{\sigma \in V} \theta_1(\sigma) + \sum_{\sigma \in V} \theta_2(\sigma) \trianglelefteq J(r_1) + J(r_2) = J(r)$
$\Rightarrow \sum_{\sigma \in V} \theta(\sigma) \trianglelefteq J(r)$
$\Leftrightarrow (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r$

■

Si J es una interpretación que satisface: $\sum_{\sigma \in V} \theta(\sigma) \trianglelefteq J(r)$
entonces la interpretación $J' := J[r_1/1][r_2/1]$ no satisface $J'(r_1) + J'(r_2) = J'(r)$, ya que: $J'(r_1) + J'(r_2) = 1 + 1 > 1 > J'(r)$. Quedando demostrado así que la equivalencia es no vale en el lema.

Supongamos que en un programa de la forma: `if` c `then` s `else` s' `fi` ya demostramos las siguientes triplas:

$$\{\ c?p\ \}\ s\ \{\ \mathbb{P}(dp) \trianglerighteq r\ \} \quad \text{y} \quad \{\neg c?p\ \}\ s'\ \{\ true\ \}$$

para algún predicado p; entonces aplicando la regla (If) la siguiente tripla es válida:

$$\{\ p\ \}\ \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else}\ s'\ \texttt{fi}\ \{\ (\mathbb{P}(dp) \trianglerighteq r) + true\ \}$$

La postcondición puede interpretarse como que el estado θ que la satisface, puede dividirse en dos partes: una que satisface dp con probabilidad de al menos r (tomando $\trianglerighteq \in \{\geq, >\}$) y otra parte en la que puede valer cualquier predicado. Sería intuitivo pensar que la desigualdad se mantiene, ya que en el peor de los casos la parte derecha de la suma probabilística ($true$) aumenta la cota inferior de la probabilidad de dp. Esta observación motiva la propiedad siguiente.

Lema 9 *Si* $\trianglerighteq \in \{\geq, >\}$ *entonces:*
$(\theta, J) \vDash (\mathbb{P}(dp) \trianglerighteq r\ +\ true) \quad \Rightarrow \quad (\theta, J) \vDash \mathbb{P}(dp) \trianglerighteq r$

Dem:

$(\theta, J) \vDash \mathbb{P}(dp) \trianglerighteq r \ + \ true$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \ \wedge \ (\theta_1, J) \vDash \mathbb{P}(dp) \trianglerighteq r \ \wedge \ (\theta_2, J) \vDash true$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \ \wedge \ \sum_{\sigma \in V} \theta_1(\sigma) \trianglerighteq J(r)$ donde: $V = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp\}$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \wedge$
$\sum_{\sigma \in V} \theta(\sigma) = \sum_{\sigma \in V} (\theta_1 + \theta_2)(\sigma) = \sum_{\sigma \in V} \theta_1(\sigma) + \sum_{\sigma \in V} \theta_2(\sigma) \geq \sum_{\sigma \in V} \theta_1(\sigma) \trianglerighteq J(r)$
$\Leftrightarrow \sum_{\sigma \in V} \theta(\sigma) \trianglerighteq J(r)$
$\Leftrightarrow (\theta, J) \vDash \mathbb{P}(dp) \trianglerighteq r$

■

Notar que el lema anterior también vale para cuando $\trianglerighteq$ es $=$ y r es 1; ya que la probabilidad total no puede exceder a 1. En la anteúltima equivalencia, la implicación $(\Leftarrow)$ se justifica tomando:
$\theta_1, \theta_2 := \theta, \theta_0$ donde θ_0 es el elemento minmal de Θ que satisface: $(\forall \sigma \in S : \theta_0(\sigma) = 0)$.

Supongamos ahora que se demostró una tripla de la forma: $\{ \ \mathbb{P}(dp) \trianglelefteq r \ \} \ s \ \{ \ \mathbb{P}(dp) \trianglelefteq r' \ \}$
aplicando la regla de escalado (Lin $\cdot$), se satisface la terna: $\{ \ \rho \cdot (\mathbb{P}(dp) \trianglelefteq r) \ \} \ s \ \{ \ \rho \cdot (p(dp) \trianglelefteq r') \ \}$
Lo que uno esperaría es que se escalen linealmente los valores de r, r' por $\rho \in (0, 1)$; eso es justamente lo que muestra el siguiente lema.

Lema 10 *Si* $\trianglelefteq \in \{<, \leq, =, \geq, >\}$ *y* $\rho \in (0, 1)$ *entonces:*
$(\theta, J) \vDash \rho \cdot (\mathbb{P}(dp) \trianglelefteq r) \quad \Leftrightarrow \quad (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq \rho \cdot r$

Dem:

$(\theta, J) \vDash \rho \cdot (\mathbb{P}(dp) \trianglelefteq r)$
$\Leftrightarrow \exists \theta' : \theta = \rho \cdot \theta' \ \wedge \ (\theta', J) \vDash \mathbb{P}(dp) \trianglelefteq r$
$\Leftrightarrow \exists \theta' : \theta = \rho \cdot \theta' \ \wedge \ \sum_{\sigma \in V} \theta'(\sigma) \trianglelefteq J(r)$ donde: $V = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp\}$
$\Leftrightarrow \exists \theta' : \theta = \rho \cdot \theta' \wedge$
$\sum_{\sigma \in V} \theta(\sigma) = \sum_{\sigma \in V} (\rho \cdot \theta')(\sigma) = \rho \cdot \sum_{\sigma \in V} \theta'(\sigma) \trianglelefteq \rho \cdot J(r)$
$\Leftrightarrow \sum_{\sigma \in V} \theta(\sigma) \trianglelefteq \rho \cdot J(r)$
$\Leftrightarrow (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq \rho \cdot r$

■

En este lema aparece una constante $\rho \in (0, 1)$; se puede generalizar la regla (Lin $\cdot$) para escalar por otro tipo de expresiones úliles, como por ejemplo: $r, \rho^i, (1 - \rho)^i$,etc. En realidad podría ser cualquier expresión e_r del conjunto $RealExp$ que satisfaga: $\mathcal{V}_r(e'_r)(\theta, J) \in (0, 1)$; excepto aquellas que involucran el constructor $\mathbb{P}(\cdots)$ para que el escalado sea independiente del valor que tengan las variables de programa (el cual está sujeto a una distribución de probabilidades). Mas formalmente, redefinimos la regla:

$$\frac{\{p\} \ s \ \{q\}}{\{e'_r \cdot p\} \ s \ \{e'_r \cdot q\}} \quad (\text{Lin}\cdot)$$

donde:

$$e'_r \ ::= \ \rho \mid r \mid e'_r + e'_r \mid e'_r - e'_r \mid e'_r * e'_r \mid e'_r / e'_r \mid {e'}^e_r$$

Además es necesario modificar ligeramente la semántica. Antes se tenía que:

$$(\theta, J) \vDash \rho \cdot p \quad \Longleftrightarrow \quad (\ \exists \theta' : \theta = \rho \cdot \theta' \wedge (\theta', J) \vDash p \ )$$

Ahora se tiene:

$$(\theta, J) \vDash \ e'_r \cdot p \quad \Longleftrightarrow \quad (\ \exists \theta' : \theta = \mathcal{V}_r(e'_r)(\theta, J) \cdot \theta' \quad \wedge \ (\theta', J) \vDash p\ )$$

Estos cambios fueron consultados a *Hartog*, la respuesta del mismo fue que no son trascendetes; ya que solo sirven para poder escalar por expresiones más generales y no alteran la validéz de la regla (Lin $\cdot$).

Una propiedad bastante evidente que no se pondrá como lema, pero que sin embargo se la usará en el siguiente lema es que si $c \in BC\langle PVar\rangle$ entonces: $\forall I \in \mathcal{I} : \mathcal{B}(c)(\sigma) \iff (\sigma, I) \vDash c$. El próximo lema esta estrechamente relacionado con el lema 7, ya que dice que significa exactamente cortar al predicado $r_1 + r_2 = r \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2$ por la condición c.

Lema 11 *Si* $c \in BC\langle PVar\rangle$ y $\trianglelefteq \in \{<, \leq, =, \geq, >\}$ *entonces:*
$(\theta, J) \vDash c?\big(r_1 + r_2 = r \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2\big) \ \Rightarrow$
$(\theta, J) \vDash r_1 + r_2 = r \wedge \mathbb{P}(dp \wedge c) \trianglelefteq r_1$

Dem:

$(\theta, J) \vDash c?\big(r_1 + r_2 = r \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2\big)$
$\Leftrightarrow \exists \theta' : \theta = c?\theta' \ \wedge \ (\theta', J) \vDash r_1 + r_2 = r \wedge \ \mathbb{P}(dp \wedge c) \trianglelefteq r_1 \ \wedge \ \mathbb{P}(dp \wedge \neg c) \trianglelefteq r_2$
$\Leftrightarrow \exists \theta' : \theta = c?\theta' \ \wedge \ J(r_1) + J(r_2) = J(r) \wedge \sum_{\sigma \in V_1} \theta'(\sigma) \trianglelefteq J(r_1) \ \wedge \sum_{\sigma \in V_2} \theta'(\sigma) \trianglelefteq J(r_2)$
donde: $V_1 = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge c\}$ y $V_2 = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge \neg c\}$
$\Rightarrow \exists \theta' : \theta = c?\theta' \ \wedge \ J(r_1) + J(r_2) = J(r) \ \wedge$
$\sum_{\sigma \in V_1} \theta(\sigma) = \sum_{\sigma \in V_1} (c?\theta')(\sigma) = \sum_{\sigma \in V'_1} \theta'(\sigma) = \sum_{\sigma \in V_1} \theta'(\sigma) \trianglelefteq J(r_1)$
Subprueba:
$V'_1 = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge c \ \wedge \ \mathcal{B}(c)(\sigma')\}$
$= \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge c \ \wedge \ (\sigma', J_{\mathcal{I}}) \vDash c\}$
$= \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp \wedge c\} \quad = \quad V_1 \quad \bullet$
$\Rightarrow (\theta, J) \vDash r_1 + r_2 = r \ \wedge \ (\theta, J) \vDash \mathbb{P}(dp \wedge c) \trianglelefteq r_1$
$\Leftrightarrow (\theta, J) \vDash r_1 + r_2 = r \wedge \mathbb{P}(dp \wedge c) \trianglelefteq r_1$

■

Lema 12 *La semántica denotacional del lenguaje* $\mathcal{L}_{pw}$ *es lineal con resprecto al estado probabilístico, es decir:*

$$\forall s \in \mathcal{L}_{pw} \ : \forall \ \theta, \theta' \in \Theta \ : \quad \mathcal{D}(s)(\rho \cdot \theta + \theta') \ = \ \rho \cdot \mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta')$$

Dem:

La demostración es por inducción estructural en los programas de $\mathcal{L}_{pw}$. Para el caso de iteración, definimos: $\texttt{if}^{\ n} := (\ \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else skip fi}\ )^n$

• **caso** `skip`
$\mathcal{D}(\ \texttt{skip}\ )(\rho \cdot \theta + \theta')$
$= \ \rho \cdot \theta + \theta'$
$= \ \rho \cdot \mathcal{D}(\ \texttt{skip}\ )(\theta) + \mathcal{D}(\ \texttt{skip}\ )(\theta')$

- **caso** $x := e$

$\mathcal{D}(x := e)(\rho \cdot \theta + \theta')$
$= \ (\rho \cdot \theta + \theta')[x/\mathcal{V}(e)]$
$= \ \rho \cdot \theta[x/\mathcal{V}(e)] \ + \ \theta'[x/\mathcal{V}(e)]$
$= \ \rho \cdot \mathcal{D}(x := e)(\theta) \ + \ \mathcal{D}(x := e)(\theta')$

- **caso** $s; s'$

$\mathcal{D}(s; s')(\rho \cdot \theta + \theta')$
$= \ \mathcal{D}(s')\mathcal{D}(s)(\rho \cdot \theta + \theta')$
$= \ \mathcal{D}(s')\big(\ \rho \cdot \mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta') \ \big) \quad \leftarrow$ h.i. sobre: s, θ, θ'
$= \ \rho \cdot \mathcal{D}(s')\mathcal{D}(s)(\theta) + \mathcal{D}(s')\mathcal{D}(s)(\theta') \quad \leftarrow$ h.i. sobre: $s', \mathcal{D}(s)(\theta), \mathcal{D}(s)(\theta')$
$= \ \rho \cdot \mathcal{D}(s; s')(\theta) + \mathcal{D}(s; s')(\theta')$

- **caso** $s \oplus_{\rho'} s'$

$\mathcal{D}(s \oplus_{\rho'} s')(\rho \cdot \theta + \theta')$
$= \ \mathcal{D}(s)(\rho \cdot \theta + \theta') \ \oplus_{\rho'} \ \mathcal{D}(s')(\rho \cdot \theta + \theta')$
$= \ \big(\ \rho \cdot \mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta') \ \big) \ \oplus_{\rho'} \ \mathcal{D}(s')(\rho \cdot \theta + \theta') \quad \leftarrow$ h.i. sobre: s, θ, θ'
$= \ \big(\ \rho \cdot \mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta') \ \big) \ \oplus_{\rho'} \ \big(\ \rho \cdot \mathcal{D}(s')(\theta) + \mathcal{D}(s')(\theta') \ \big) \quad \leftarrow$ h.i. sobre: s', θ, θ'
$= \ \rho' \cdot \big(\ \rho \cdot \mathcal{D}(s)(\theta) + \mathcal{D}(s)(\theta') \ \big) \ + \ (1 - \rho') \cdot \big(\ \rho \cdot \mathcal{D}(s')(\theta) + \mathcal{D}(s')(\theta') \ \big)$
$= \ \rho \cdot \rho' \cdot \mathcal{D}(s)(\theta) \ + \ \rho \cdot (1 - \rho') \cdot \mathcal{D}(s')(\theta) \ + \ \rho' \cdot \mathcal{D}(s)(\theta') \ + \ (1 - \rho') \cdot \mathcal{D}(s')(\theta')$
$= \ \rho \cdot \big(\ \rho' \mathcal{D}(s)(\theta) + (1 - \rho') \cdot \mathcal{D}(s)(\theta) \ \big) \ + \big(\ \rho' \cdot \mathcal{D}(s)(\theta') \ + \ (1 - \rho') \cdot \mathcal{D}(s')(\theta') \ \big)$
$= \ \rho \cdot \mathcal{D}(s \oplus_{\rho'} s')(\theta) \ + \ \mathcal{D}(s \oplus_{\rho'} s')(\theta')$

- **caso** `if` c `then` s `else` s' `fi`

$\mathcal{D}(\ \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else}\ s'\ \texttt{fi}\ )(\rho \cdot \theta + \theta')$
$= \ \mathcal{D}(s)(c?(\rho \cdot \theta + \theta')) \ + \ \mathcal{D}(s')(\neg c?(\rho \cdot \theta + \theta'))$
$= \ \mathcal{D}(s)(\rho \cdot (c?\theta) + (c?\theta')) \ + \ \mathcal{D}(s')(\rho \cdot (\neg c?\theta) + (\neg c?\theta'))$
$= \ \rho \cdot \mathcal{D}(s)(c?\theta) \ + \ \mathcal{D}(s)(c?\theta') \ + \ \mathcal{D}(s')(\rho \cdot (\neg c?\theta) + (\neg c?\theta')) \quad \leftarrow$ h.i. sobre: $s, (c?\theta), (c?\theta')$
$= \ \rho \cdot \mathcal{D}(s)(c?\theta) \ + \ \mathcal{D}(s)(c?\theta') \ + \ \rho \cdot \mathcal{D}(s')(\neg c?\theta) \ + \ \mathcal{D}(s')(\neg c?\theta') \quad \leftarrow$ h.i. sobre: $s', (\neg c?\theta), (\neg c?\theta')$
$= \ \rho \cdot \big(\ \mathcal{D}(s)(c?\theta) \ + \ \mathcal{D}(s')(\neg c?\theta) \ \big) \ + \big(\ \mathcal{D}(s)(c?\theta') \ + \ \mathcal{D}(s')(\neg c?\theta') \ \big)$
$= \ \rho \cdot \mathcal{D}(\ \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else}\ s'\ \texttt{fi}\ )(\theta) \ + \ \mathcal{D}(\ \texttt{if}\ c\ \texttt{then}\ s\ \texttt{else}\ s'\ \texttt{fi}\ )(\theta')$

- **caso** `while` c `do` s `od`

$\mathcal{D}(\ \texttt{while}\ c\ \texttt{do}\ s\ \texttt{od}\ )(\rho \cdot \theta + \theta')$
$= \ \lim_{n \to \infty} \neg c \ ? \ \mathcal{D}(\ \texttt{if}\ ^n)(\rho \cdot \theta + \theta')$
$= \ \lim_{n \to \infty} \neg c \ ? \ \big(\ \rho \cdot \mathcal{D}(\ \texttt{if}\ ^n)(\theta) \ + \ \mathcal{D}(\ \texttt{if}\ ^n)(\theta') \ \big) \quad \leftarrow$ h.i. sobre: $\texttt{if}\ ^n, \theta, \theta'$
$= \ \lim_{n \to \infty} \rho \cdot \big(\neg c \ ? \ \mathcal{D}(\ \texttt{if}\ ^n)(\theta) \ \big) \ + \ \big(\neg c \ ? \ \mathcal{D}(\ \texttt{if}\ ^n)(\theta') \ \big)$
$= \ \rho \cdot \lim_{n \to \infty} \neg c \ ? \ \mathcal{D}(\ \texttt{if}\ ^n)(\theta) \ + \ \lim_{n \to \infty} \neg c \ ? \ \mathcal{D}(\ \texttt{if}\ ^n)(\theta')$
$= \ \rho \cdot \mathcal{D}(\ \texttt{while}\ c\ \texttt{do}\ s\ \texttt{od}\ )(\theta) \ + \ \mathcal{D}(\ \texttt{while}\ c\ \texttt{do}\ s\ \texttt{od}\ )(\theta')$

■

El próximo lema se usará en algunas verificaciones de fragmentos `if` ; básicamente muestra como la ecuación $\theta + \theta_0 = \theta$ se refleja en los predicados probabilísticos que invoculcran el constructor $\mathbb{P}(\cdots)$.

Lema 13 *Si* $\trianglelefteq \in \{<, \leq, =, \geq, >\}$ *entonces:*
$(\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r + \mathbb{P}(true) = 0 \quad \Leftrightarrow \quad (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r$

Dem:

$(\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r + \mathbb{P}(true) = 0$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \ \wedge (\theta_1, J) \vDash \mathbb{P}(dp) \trianglelefteq r \ \wedge \ (\theta_2, J) \vDash \mathbb{P}(true) = 0$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \ \wedge \sum_{\sigma \in V} \theta_1(\sigma) \trianglelefteq J(r) \wedge \sum_{\sigma \in S} \theta_2(\sigma) = 0$
donde: $V = \{\sigma' \mid (\sigma', J_{\mathcal{I}}) \vDash dp\}$
$\Leftrightarrow \exists \theta_1, \theta_2 : \theta_1 + \theta_2 = \theta \ \wedge \sum_{\sigma \in V} \theta(\sigma) = \sum_{\sigma \in V} \theta_1(\sigma) + \sum_{\sigma \in V} \theta_2(\sigma) \trianglelefteq J(r) + 0 = J(r)$
$\Leftrightarrow \sum_{\sigma \in V} \theta(\sigma) \trianglelefteq J(r)$
$\Leftrightarrow (\theta, J) \vDash \mathbb{P}(dp) \trianglelefteq r + \mathbb{P}(true) = 0$

■

Notar que en la segunda equivalencia del lema anterior se usó que cualquier estado determinístico $\sigma \in S$ satisface *true*. Por otra parte la implicación $(\Leftarrow)$ de la cuarta equivalencia se obtiene definiendo: $\theta_1, \theta_2 := \theta, \theta_0$ recordando que θ_0 es el menor elemento de Θ.

Apéndice B

Propiedades de la lógica de Morgan

En este apéndice se justificarán varias propiedades que fueron necesarias para verificar el algoritmo de $MillerRabin$ usando la extensión de *Morgan*. La mayoría de ellas están extraidas de [5], [7].

Lema 14 (Sub-linearidad) *Sean $a, b, c \in \mathbb{R}_{\geq}$ reales no negativos; $Q, Q' \in \mathcal{P}(St)$ predicados probabilísticos y S un programa PCGL entonces:*

$$wp.S.(a * Q + b * Q' \ \ominus c) \quad \Leftarrow \quad a * wp.S.Q + b * wp.S.Q' \ \ominus c$$

Dem:

Todo el desarrollo que justifica la propiedad de *sub-linearidad* se encuentra en [5].

■

Notar que tomando $a, b, c := 1, 1, 0$; se deriva una propiedad muy utilizada en la verificación de $MillerRabin$: $wp.S.(Q + Q') \ \Leftarrow \ wp.S.Q + wp.S.Q'$

La siguiente propiedad es la monotonía probabilística; que generaliza la monotonía estandard del mismo modo que la relación $\Rrightarrow$ generaliza la implicación $\Rightarrow$.

Lema 15 (Monotonía) *Sean $Q, Q' \in \mathcal{P}(St)$ predicados probabilísticos tales que $Q \Rrightarrow Q'$ y S un programa PCGL entonces:*

$$wp.S.Q \ \Rrightarrow \ wp.S.Q$$

Dem:

$wp.S.Q$
$\equiv$ { algebra }
$wp.S.((Q - Q') + Q')$
$\Leftarrow$ { sub-linearidad: $a, b, c := 1, 1, 0$ }
$wp.S.(Q - Q') \ + \ wp.S.Q'$
$\Leftarrow$ { $0 \Leftarrow wp.S.(Q - Q')$ }
$wp.S.Q'$

■

Otra propiedad que es consecuencia de la monotonía, y que se utilizó varias veces en la verificación del algoritmo es la *sub-distributividad* del operador $\sqcup$.

$$\left.\begin{array}{lcl} Q & \Rrightarrow & Q \sqcup Q' \\ Q' & \Rrightarrow & Q \sqcup Q' \end{array}\right\} \Rightarrow \left.\begin{array}{lcl} wp.S.Q & \Rrightarrow & wp.S.(Q \sqcup Q') \\ wp.S.Q' & \Rrightarrow & wp.S.(Q \sqcup Q') \end{array}\right\} \Rightarrow \quad wp.S.Q \sqcup wp.S.Q' \;\Rrightarrow\; wp.S.(Q \sqcup Q')$$

El siguiente resultado expresa que las precondiciones probabilísticas están acotadas superiormente. En la siguiente propiedad se denotará $Q_{max} := \bigsqcup_{s \in St} Q.s$, representando el máximo Q sobre todos los valores de las variables de programa.

Lema 16 (Factibilidad) *Sea $Q \in \mathcal{P}(St)$ un predicado probabilístico y S un programa PCGL entonces:*

$$wp.S.Q \;\Rrightarrow\; Q_{max}$$

Dem:

En primer lugar se observa que:
$wp.S.0$
$\equiv$ { algebra }
$wp.S.(2 * 0)$
$\Leftarrow$ { sub-linearidad: $a, b, c := 2, 0, 0$ }
$2 * wp.S.0$
por lo tanto vale que $wp.S.0 \equiv 0$. Luego:

0
$\equiv$ { resultado anterior }
$wp.S.0$
$\equiv$ { $0 \;\equiv\; Q \ominus Q_{max}$ }
$wp.S.(\; Q \ominus Q_{max} \;)$
$\Leftarrow$ { sub-linearidad: $a, b, c := 1, 0, Q_{max}$ }
$wp.S.Q \;\ominus\; Q_{max}$
Finalmente sumando Q_{max} en ambos extremos se obtiene que:

$$wp.S.Q \quad \Rrightarrow \quad Q_{max}$$

■

La siguiente propiedad expresa que la multiplicación por un escalar c se distribuye con repecto a los comandos *PCGL*.

Lema 17 (Escalado) *Sea $Q \in \mathcal{P}(St)$ un predicado probabilístico; $c \in \mathbb{R}_{\geq}$ una constante real; y S un programa PCGL entonces:*

$$wp.S.(c * Q) \quad \equiv \quad c * wp.S.Q$$

Dem:

Separamos la prueba en dos casos:

- **caso** $c = 0$

$$
\begin{array}{ll}
& wp.S.(0 * Q) \\
\equiv & \{ \text{ algebra } \} \\
& wp.S.0 \\
\equiv & \{ \text{ factibilidad } \} \\
& 0 \\
\equiv & \{ \text{ algebra } \} \\
& 0 * wp.S.Q
\end{array}
$$

- **caso** $c \neq 0$

$$
\begin{array}{ll}
& wp.S.(c * Q) \\
\equiv & \{ \ c \neq 0 \ ; \text{ algebra } \} \\
& c * \frac{1}{c} * wp.S.(c * Q) \\
\Rrightarrow & \{ \text{ sub-linearidad } \} \\
& c * (wp.S.(\frac{1}{c} * c * Q)) \\
\equiv & \{ \ c \neq 0 \ ; \text{ algebra } \} \\
& c * wp.S.Q
\end{array}
$$

Usando sub-linearidad se obtiene la otra implicación probabilística:

$wp.S.(c * Q) \ \Lleftarrow \ c * wp.S.Q.$

Por lo tanto: $wp.S.(c * Q) \ \equiv \ c * wp.S.Q.$

■

La próxima propiedad tiene que ver con la conjunción probabilística & que también satisface las propiedades que son consistentes con los booleanos:

$$0\&0 = 0 \qquad 0\&1 = 0 \qquad 1\&0 = 0 \qquad 1\&1 = 1$$

Lema 18 (Sub-conjuntividad) *Sean $Q, Q' \in \mathcal{P}(St)$ predicados probabilísticos y S un programa PCGL entonces:*

$$wp.S.(Q\&Q') \quad \Lleftarrow \quad wp.S.Q \ \& \ wp.S.Q'$$

Dem:

$$
\begin{array}{ll}
& wp.S.(Q\&Q') \\
\equiv & \{ \text{ definición de } \& \ \} \\
& wp.S.((Q + Q') \ominus 1) \\
\Lleftarrow & \{ \text{ sub-linearidad: } a, b, c := 1, 1, 1 \ \} \\
& wp.S.Q + wp.S.Q' \ \ominus \ 1 \\
\equiv & \{ \text{ definición de } \& \ \} \\
& wp.S.Q \ \& \ wp.S.Q'
\end{array}
$$

■

De esta última propiedad se puede extraer otra menos general, que se usará mucho en la verificación de $MillerRabin$. En primer lugar observamos que & se reduce a $\sqcap$ cuando alguno de los operandos es estandard (digamos $[P]$):

$$Q\&[P] \quad \equiv \quad (Q + [P]) \ominus 1 \quad \equiv \quad (Q + [P] - 1) \sqcup 0 \quad \equiv \quad Q \sqcap [P]$$

Por lo tanto si el predicado probabilístico $wp.S.[P]$ también es estandard entonces se tiene una sub-distributividad para el operador $\sqcap$:

$$wp.S.(Q \sqcap [P]) \quad \Leftarrow \quad wp.S.Q \ \sqcap \ wp.S.[P]$$

Notar además que si S en un programa estandard (que no involucra el operador $\oplus_p$) y $[P], [P']$ son predicados estandard entonces, se tiene una propiedad de *conjuntividad-total* como se muestra a continuación.

$wp.S.[P]$ & $wp.S.[P']$
$\equiv$ { S es estandard }
$wp.S.[P] \ \sqcap \ wp.S.[P']$
$\Leftarrow$ { $[P] \Leftarrow [P]\&[P']$; monotonía }
$wp.S.([P]\&[P']) \ \sqcap \ wp.S.[P']$
$\Leftarrow$ { $[P'] \Leftarrow [P]\&[P']$; monotonía }
$wp.S.([P]\&[P']) \ \sqcap \ wp.S.([P]\&[P'])$
$\equiv$ { $\sqcap$ es idempotente }
$wp.S.([P]\&[P'])$

Tomando $Q, Q' := [P], [P']$ en la propiedad de *sub-conjuntividad* se tiene:

$$wp.S.[P] \ \& \ wp.S.[P'] \quad \Rrightarrow \quad wp.S.([P]\&[P'])$$

Por lo tanto:

$$wp.S.[P] \ \& \ wp.S.[P'] \quad \equiv \quad wp.S.([P]\&[P'])$$

Bibliografía

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

[2] Joe Hurd. *Verification of the Miller-Rabin probabilistic primality test.* Journal of Logic and Algebraic Programming, 50(1-2):3-21, May-August 2003. Special issue on Probabilistic Techniques for the Design and Analysis of Systems.

[3] Joe Hurd *Formal Verification of probabilistic algorithms.* PhD thesis, University of Cambridge, 2002.

[4] J.I. Den Hartog and E.P. De Vink. *Verifying probabilistic programs using a Hoare like logic.* In P.S. Thiagarajan and R. Yap, editors, LNCS 1742 (ASIAN'99), 113-125. Springer 1999.

[5] Carroll Morgan, Annabelle McIver, Karen Seidel and J. W. Sanders. *Probabilistic predicate transformers.* ACM Transaction on Programming Languages and Systems, 18(3): 325-353, 1996.

[6] Carroll Morgan. *Proof rules for probabilistic loops.* In Proceedings of the BCS-FACS 7th Refinement Workshop, 1996.

[7] Carroll Morgan and Annabelle McIver. *pCGL: Formal reasoning for random algorithms* South African Computer Journal, (22):14-27, March 1999. Special Issue on the 1999 Winter School on Formal and Applied Compuer Science.

[8] Karen Seidel, Carroll Morgan and Annabelle McIver. *Probabilístic imperative programming: A rigorous approach* In L. Groves and S. Reeves, editors, Formal Methods Pacific '97. Springer-Verlag, 1997.

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

[10] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[11] Calin Fidge and Carron Shankland. *But what if I don't want to wait for ever?* Formal Aspects of Computing, to appear in 2003.

[12] G.L.Miller. *Riemman's hypothesis and tests for primality* Conference Record of Seventh Annual ACM Symposium on Theory of Computation, Albuquerque, New Mexico, May 1975, pp. 234-239.

[13] M.O.Miller. *Probabilistic algorithms* J.F. Trub (Ed.), Algorithms and Complexity: New Directions and Rcenet Results, Academic Press, New York 1976, pp 21-39.