Capítulo 11
Inducción y recursión

Índice del Capítulo

11.1. Introducción

Una técnica muy utilizada para demostrar propiedades sobre el conjunto de números naturales, es el *principio de inducción*. Este principio provee una herramienta poderosa y fácil de usar. La idea de las demostraciones por inducción se basa en lo siguiente: puesto que sumando 1 al natural k se obtiene $k+1$ que es mayor que k, no existe ningún natural que sea el *mayor* de todos. Sin embargo, partiendo del número 1 se pueden alcanzar todos los enteros positivos, después de un número finito de pasos, pasando sucesivamente de k a $k+1$. Supongamos entonces una cierta propiedad $P(n)$ que sabemos cierta para $n = 1$, luego si la propiedad es válida para un entero en particular y de esto se concluye que también será válida para el entero siguiente, entonces se podrá afirmar que resultará cierta para todos los enteros positivos.

La idea de inducción se puede ilustrar de muchas maneras no matemáticas. Por ejemplo, consideremos una fila de fichas de dominó ubicadas en posición vertical y numeradas, de modo tal que, si se cae hacia atrás la ficha situada en la posición k hará caer a la que está detrás, en la posición $k+1$. En seguida, se puede intuir lo que occurrirá si la ficha situada en la posición 1 se

cae hacia atrás. También, es claro que si fuera empujada hacia atrás la ficha en la posición n_1 todas las fichas ubicadas por detrás de ésta caerían. Este ejemplo ilustra una ligera generalización del principio de inducción.

11.2. Inducción matemática

Consideremos el siguiente predicado:

$$P.n : (\sum i : 1 \leq i \leq n : 2 \times i - 1) = n^2 \tag{11.1}$$

Por ejemplo, para n igual a 2 y 3 respectivamente, 11.1 establece que $1+3 = 2^2$ y $1+3+5 = 3^2$, lo cual indica que $P.2$ y $P.3$ resultan *true*. ¿Será $P.n$ cierto siempre?, o bien ¿es verdera la cuantificación $(\forall\, n : \mathbb{N} : 0 \leq n : P.n)$?

Una forma de probar que P es cierta para todo número natural es demostrar que $P.0$ es cierta y que suponiendo que P es cierta para todos los naturales menores o iguales a un natural n dado, entonces también es cierta $P.(n+1)$. Esta última afirmación puede escribirse como:

$$(\forall\, n : \mathbb{N} : 0 < n : P.0 \wedge P.1 \wedge \ldots \wedge P(n-1) \Rightarrow P.n) \tag{11.2}$$

Ahora, si ambas cosas son ciertas, podemos ver cómo construir una demostración de P para un número N dado:

- La demostración de $P.0$ ya está hecha.
- De la veracidad de $P.0$ y $P.0 \Rightarrow P.1$ (que no es más que (11.2) en la instancia $n := 1$), y usando Modus Ponens (3.77) concluímos que $P.1$ es verdadera.
- De la veracidad de $P.0 \wedge P.1$ y $P.0 \wedge P.1 \Rightarrow P.2$ (que no es más que (11.2) en la instancia $n := 2$), y usando Modus Ponens (3.77) concluímos que $P.2$ es verdadera.
- ...
- De la veracidad de $P.0 \wedge \ldots \wedge P.(N-1)$ y $P.0 \wedge \ldots \wedge P.(N-1) \Rightarrow P.N$ (que no es más que (11.2) en la instancia $n := N$), y usando Modus Ponens (3.77) concluímos que $P.N$ es verdadera.

Por supuesto, no necesitamos hacer todo esto para demostrar que $P.N$ es verdadera, es suficiente saber que en principio, podemos hacer esto para probarlo.

El principio de *inducción matemática*, permite inferir a partir de las pruebas de $P.0$ y (11.2) que $P.n$ es verdadera para cualquier natural n.

Consideremos de nuevo el ejemplo. Demostraremos primero $P.0$ (llamado *caso base*).

Caso base

$$
\begin{array}{ll}
 & (\sum\, i : 1 \le i \le 0 : 2 \times i - 1) \\
= & \quad \langle \text{neutro de } + \text{ (debido a que el rango es vacío) } 6.4 \rangle \\
 & 0 \\
= & \quad \langle \text{aritmética} \rangle \\
 & 0^2
\end{array}
$$

Demostraremos ahora el *paso inductivo*, suponiendo que $P.0$, $P.1$, .. y $P.n$ son ciertas demostraremos que $P.(n+1)$ es cierta.

Paso inductivo

$$
\begin{array}{ll}
 & (\sum\, i : 1 \le i \le n+1 : 2 \times i - 1) \\
= & \quad \langle \text{Separación de término } (6.14) \rangle \\
 & (\sum\, i : 1 \le i \le n : 2 \times i - 1) + 2 \times (n+1) - 1 \\
= & \quad \langle \text{Hipótesis inductiva } P.n \rangle \\
 & n^2 + 2 \times (n+1) - 1 \\
= & \quad \langle \text{Aritmética } \rangle \\
 & (n+1)^2
\end{array}
$$

La prueba anterior, emplea una técnica que se utiliza a menudo: el lado izquierdo de $P(n+1)$ es manipulado de forma tal de "exponer" $P.n$, es decir, hacer posible el uso de la hipótesis de inducción $P.n$. Aquí, con la separación de término, se puso en evidencia a $P.n$.

El principio de inducción se formaliza en un axioma de nuestro cálculo de predicados como sigue, donde $P : \mathbb{N} \to \mathbb{B}$:

(11.1) Axioma. Inducción Matemática sobre $\mathbb{N}$:

$$(\forall\, n : \mathbb{N} :: (\forall\, i : 0 \le i < n : P.i) \Rightarrow P.n) \Rightarrow (\forall\, n : \mathbb{N} :: P.n).$$

El consecuente en el axioma 11.1 trivialmente implica el antecedente, con lo cual podemos reescribir el axioma anterior utilizando Implicación mutua (3.80) de la siguiente manera:

(11.2) Teorema. Inducción Matemática sobre $\mathbb{N}$:

$$(\forall\, n : \mathbb{N} :: (\forall\, i : 0 \le i < n : P.i) \Rightarrow P.n) \equiv (\forall\, n : \mathbb{N} :: P.n).$$

En general utilizaremos el teorema 11.2 en lugar del axioma 11.1 para probar propiedades por inducción, dado que este teorema es más fácil de usar por ser el operador $\equiv$ simétrico, mientras que el operador $\Rightarrow$ no lo es.

El caso $P.0$ está incluido en el axioma 11.1. Veremos ahora que al manipular el antecedente el caso $P.0$ aparece.

$$
\begin{array}{ll}
 & (\forall\, n : 0 \le n : (\forall\, i : 0 \le i < n : P.i) \Rightarrow P.n) \\
= & \quad \langle \text{Separación de término } 6.14 \rangle \\
 & ((\forall\, i : 0 \le i < 0 : P.i) \Rightarrow P.0) \wedge (\forall\, n : 1 \le n : (\forall\, i : 0 \le i < n : P.i) \Rightarrow P.n) \\
= & \quad \langle \text{Rango vacío } 6.4 \rangle \\
 & (\mathit{true} \Rightarrow P.0) \wedge (\forall\, n : 1 \le n : (\forall\, i : 0 \le i < n : P.i) \Rightarrow P.n) \\
= & \quad \langle \text{Neutro a izquierda de } \Rightarrow \ 3.73\text{; Cambio de dummy } 6.15\text{; artimética} \rangle \\
 & P.0 \wedge (\forall\, n : 0 \le n : (\forall\, i : 0 \le i < n+1 : P.i) \Rightarrow P.(n+1))
\end{array}
$$

Entonces, podemos reescribir el axioma 11.1 de la siguiente forma, que es la forma que generalmente se usa cuando es necesario probar propiedades mediante inducción:

(11.3) Teorema. Inducción Matemática sobre $\mathbb{N}$:

$$P.0 \wedge (\forall\, n : \mathbb{N} : (\forall\, i : 0 \leq i \leq n : P.i) \Rightarrow P(n+1)) \equiv (\forall\, n : \mathbb{N} :: P.n).$$

La primera conjunción en el axioma 11.3 es el *caso base* de la inducción matemática, la segunda conjunción del antecedente:

$$(\forall\, n : \mathbb{N} : (\forall\, i : 0 \leq i \leq n : P.i) \Rightarrow P(n+1)) \tag{11.3}$$

es el *paso inductivo*, y $(\forall\, i : 0 \leq i \leq n : P.i)$ se llama *hipótesis de inducción*.

Para probar $(\forall\, n : \mathbb{N} :: P.n)$ por inducción, probamos el caso base y el paso inductivo por separado, y luego aseguramos en lenguaje corriente, que $P.n$ se satisface para todo número natural n. La prueba del paso inductivo, se hace probando $(\forall\, i : 0 \leq i \leq n : P.i) \Rightarrow P(n+1)$ para un $n \geq 0$ arbitrario. Más aún, $(\forall\, i : 0 \leq i \leq n : P.i) \Rightarrow P(n+1)$ se prueba usualmente suponiendo $(\forall\, i : 0 \leq i \leq n : P.i)$ y luego probando $P(n+1)$.

El axioma (11.1) y los teoremas (11.2) y (11.3), describen la técnica de inducción sobre todos los números naturales, sin embargo, la inducción puede realizarse sobre cualquier subconjunto $n_0, n_0+1, n_0+2, \ldots$ de $\mathbb{N}$. La única diferencia es el punto de partida, el caso base será $P.n_0$. Reformulamos entonces el principio de inducción matemática para estos casos del siguiente modo:

(11.4) Teorema. Inducción matemática sobre $\{n_0, n_0+1, \ldots\}$:

$$P.n_0 \wedge (\forall\, n : n_0 \leq n : (\forall\, i : n_0 \leq i \leq n : P.i) \Rightarrow P(n+1)) \equiv (\forall\, n : n_0 \leq n : P.n).$$

Veamos un ejemplo,

(11.5) Ejemplo. En este caso, queremos probar $(\forall\, n : 3 \leq n : P.n)$ donde

$$P.n : 2 \times n + 1 < 2^n.$$

Caso base

$P.3$ es $2 \times 3 + 1 < 2^3$, lo cual es cierto.

Paso inductivo

Para un valor arbitrario $n \geq 3$, probaremos $P.(n+1)$ usando la hipótesis de inducción $P.n$.

$$
\begin{array}{ll}
 & 2^{n+1} \\
= & \quad \langle \text{aritmética} \rangle \\
 & 2 \times 2^{n} \\
> & \quad \langle \text{hipótesis de inducción } P.n \rangle \\
 & 2 \times (2 \times n + 1) \\
= & \quad \langle \text{aritmética} \rangle \\
 & 2 \times (n + 1) + 1 + 2 \times n - 1 \\
> & \quad \langle \text{aritmética } 2 \times n - 1 > 0, \text{ ya que } 3 \leq n \rangle \\
 & 2 \times (n + 1) + 1
\end{array}
$$

Aquí, para probar el paso inductivo, transformamos el lado derecho de $P.(n + 1)$ en el lado izquierdo. En vez de esto, podríamos haber transformado el lado izquierdo en el derecho o también $P.(n + 1)$ en *true*.

Nuestro siguiente ejemplo, concierne a los números de *Fibonacci*. Estos números son interesantes cuando trabajamos con inducción matemática, pero son muy interesantes también por sí solos. Fueron introducidos por Leonardo Fibonacci (1202) a quien se le debe el nombre.

Los primeros 8 números de Fibonacci son 0, 1, 1, 2, 3, 5, 8, 13, y a excepción de los primeros dos, los siguientes números se consiguen sumando los dos anteriores. Los números de Fibonacci, se definen recursivamente del siguiente modo:

$$
\begin{array}{rcl}
fib.0 & \doteq & 0 \\
fib.1 & \doteq & 1 \\
fib.(n + 2) & \doteq & fib.(n + 1) + fib.n
\end{array}
$$

Como ya dijimos, esta sucesión de números tiene propiedades muy interesantes, algunas de las cuales se presentan en los ejercicios. Vamos a demostrar ahora, una muy simple:

$$(\forall\, n : 0 \leq n : fib.n < 2^n)$$

Dada la definición de fib, parece conveniente considerar dos casos base, cuando $n = 0$ y cuando $n = 1$. El paso inductivo podrá considerarase entonces para $n \geq 2$.

Caso base

$$
\begin{array}{ll}
 & fib.0 \\
= & \quad \langle \text{definición de } fib \rangle \\
 & 0 \\
< & \quad \langle 0 < 1 \rangle \\
 & 1 \\
= & \quad \left\langle 2^0 = 1 \right\rangle \\
 & 2^0
\end{array}
$$

El caso base cuando $n = 1$ es similar al anterior.

Paso inductivo

Sea $P.n : fib.n < 2^n$, vamos a partir de $fib.(n+2)$ y probaremos que es menor que $2^{(n+2)}$

$$
\begin{array}{ll}
 & fib.(n+2) \\
= & \quad \langle \text{definición de } fib \rangle \\
 & fib.n + fib.(n+1) \\
< & \quad \langle \text{hipótesis inductiva } P.n \rangle \\
 & 2^n + fib.(n+1) \\
< & \quad \langle \text{hipótesis inductiva } P.(n+1) \rangle \\
 & 2^n + 2^{(n+1)} \\
< & \quad \langle \text{la función } 2^x \text{ es creciente} \rangle \\
 & 2^{(n+1)} + 2^{(n+1)} \\
= & \quad \langle \text{propiedad de la función exponencial} \rangle \\
 & 2^{(n+2)}
\end{array}
$$

11.3. Ayudas para pruebas por inducción

El primer paso para probar una cuantificación universal por inducción es escribir la fórmula de la forma

$$(\forall\, n : n_0 \leq n : P.n)$$

Esto significa identificar n_0 y también la propiedad a probar: $P.n$, la cual tiene que ser de tipo booleano.

Por lo general, el paso inductivo

$$(\forall\, n : n_0 \leq n : (\forall\, i : n_0 \leq i \leq n : P.i) \Rightarrow P.(n+1))$$

se realiza probando $(\forall\, i : n_0 \leq i \leq n : P.i) \Rightarrow P.(n+1)$ para un arbitrario $n \geq n_0$. Y esto último, se hace suponiendo $P.n_0, \ldots, P.n$ y probando $P.(n+1)$. Esta clase de prueba, a menudo exige la manipulación de $P.(n+1)$ de alguna manera.

El objetivo de manipular $P.(n+1)$ es hacer posible el uso de las conjunciones $P.\,0, \ldots, P.n$ en la hipótesis inductiva, es decir poner en evidencia la hipótesis inductiva. Por ejemplo, en (11.5) reescribimos $2^{(n+1)}$ como 2×2^n, de modo de poder usar $P.n : 2 \times n + 1 < 2^n$. Mientras que en (11.1), obtuvimos $P.n$ mediante separación de término. En estos ejemplos pudimos probar $P.(n+1)$ usando solo la hipótesis $P.n$, pero en otros casos necesitaremos más de una hipótesis inductiva para probar $P.(n+1)$.

11.4. Verificación de programas

Utilizaremos el principio de inducción para comprobar que un programa (que en la programación funcional sería una definición de función) satisface su especificación. A ésto se lo llama *verificación de programas*.

También utilizaremos inducción para construir una definición de función a partir de su especificación. A esta técnica se la llama *derivación de programas*, y la veremos en el capítulo siguiente.

Si bien la técnica de verificación de programas es muy útil, ya que nos permite probar que un programa es correcto según su especificación, por lo general no es una tarea trivial, por lo que es conveniente realizarla, al menos en parte, a medida que se construye el programa. Esta construcción conjunta de programa y demostración tiene la ventaja adicional de que la misma nos va guiando en la construcción del programa.

Veamos un ejemplo en el que verificamos que la función $f : Nat \to Nat$ definida recursivamente como:

$$\begin{array}{rcl} f.0 & \doteq & 0 \\ f.(i+1) & \doteq & f.i + 2 \times i + 1 \end{array}$$

satisface la siguiente especificación:

$$\begin{array}{ll} pre: & true \\ post: & f.i = i^2 \end{array}$$

Primero definimos la propiedad a probar ($P.n$) a partir de las expresiones booleanas *pre* y *post*. Dado que la precondición es *true*, $P.n$ resulta:

$$P.n : f.n = n^2$$

Caso base

$$\begin{array}{ll} & f.0 \\ = & \quad \langle \text{definición de } f \rangle \\ & 0 \\ = & \quad \langle \text{aritmética} \rangle \\ & 0^2 \end{array}$$

Paso inductivo

Probaremos $P.(n+1)$ suponiendo la hipótesis de inducción $P.n$:

$$\begin{array}{ll} & f.(n+1) \\ = & \quad \langle \text{definición de } f \rangle \\ & f.n + 2 \times n + 1 \\ = & \quad \langle \text{hipótesis inductiva} \rangle \\ & n^2 + 2 \times n + 1 \\ = & \quad \langle \text{aritmética} \rangle \\ & (n+1)^2 \end{array}$$

De esta demostración concluímos que f satisface $(\forall\, n : \mathbb{N} :: f.n = n^2)$.

Veamos otro ejemplo de verificación de programas: sea fac la función que calcula el factorial de un número, que definimos recursivamente como:

$$\begin{array}{lcl} fac.0 & \doteq & 1 \\ fac.(n+1) & \doteq & (n+1) \times fact.n \end{array}$$

Vamos a probar que $fact$ satisface la siguiente especificación:

$$\begin{array}{ll} pre: & true \\ post: & fact.n = (\prod\ i : 1 \leq i \leq n : i) \end{array}$$

A partir de esta especificación, la propiedad a probar $P.n$ resulta:

$$P.n : fact.n = (\prod\!\!\prod\ i : 1 \leq i \leq n : i)$$

Caso base $P.0$

$$\begin{array}{ll} & fact.0 \\ = & \quad \langle \text{definición de factorial} \rangle \\ & 1 \\ = & \quad \langle \text{Rango Vacío para } \prod \rangle \\ & (\prod\ i : false : i) \\ = & \quad \langle \text{Aritmética en el Rango } false \equiv 1 \leq i \leq 0 \rangle \\ & (\prod\ i : 1 \leq i \leq 0 : i) \end{array}$$

Paso inductivo

Probaremos $P.(n+1)$ suponiendo la hipótesis de inducción $P.n$:

$$\begin{array}{ll} & fact.(n+1) \\ = & \quad \langle \text{definición de factorial} \rangle \\ & (n+1) \times fact.n \\ = & \quad \langle \text{hipótesis inductiva} \rangle \\ & (n+1) \times (\prod\ i : 1 \leq i \leq n : i) \\ = & \quad \langle \text{Separación de término} \rangle \\ & (\prod\ i : 1 \leq i \leq n+1 : i) \end{array}$$

Por lo tanto hemos probado que la definición dada de $fact$ satisface la especificación dada.

11.5. Inducción sobre Listas

El principio de inducción puede también aplicarse a listas. Cuando aplicamos inducción sobre los números naturales, discriminamos dos casos: cuando el número es 0 y cuando es de la forma $n+1$ para algún natural n. De modo similar la inducción sobre listas se basa en dos casos: o bien la lista es vacía o bien es de la forma $x \triangleright xs$, para algún valor x y lista xs.

Por lo tanto, para demostrar por inducción que una propiedad $P.xs$ se cumple para cualquier lista xs, es suficiente demostrar que:

Caso base: $P.[\,]$ se satisface, y

Paso inductivo: Si $P.xs$ se satisface, entonces $P.(x \triangleright xs)$ vale para cualquier x.

Esto es válido por el mismo argumento que ya hemos utilizado para inducción sobre números naturales. Sabemos por el caso base que $P.[\,]$ se satisface, y entonces aplicando el paso inductivo $P.[x]$ también vale para cualquier x, (ya que $[x] = x \triangleright [\,]$), nuevamente, aplicando el paso inductivo $P.[y, x]$ vale para cualquier x e y, (ya que $[y, x] = y \triangleright [x]$), y así sucesivamente. Entonces $P.xs$ vale para cualquier lista finita xs. Es fácil formalizar esta prueba por inducción sobre el tamaño de la lista xs, con lo cual el principio de inducción sobre listas, es una consecuencia directa del principio de inducción sobre los números naturales.

Al igual que en la inducción sobre naturales, en la inducción sobre listas el caso base puede ser una lista con n_0 elementos. En tal caso la hipótesis inductiva es $P.xs$, donde $\#xs \geq n_0$. Por ejemplo, el principio de inducción puede ser reformulado para listas con al menos un elemento como:

Caso base: $P.[x]$ se satisface para cualquier x, y

Paso inductivo: Si $P.(x \triangleright xs)$ se satisface, entonces $P.(x' \triangleright x \triangleright xs)$ vale para cualquier x'.

Aquí concluímos que $P.xs$ vale para cualquier lista finita con al menos un elemento.

En el Capítulo 8, enunciamos diversas propiedades sobre los operadores definidos sobre listas sin dar una demostración explícita. Vamos a probar ahora algunas de las propiedades citadas.

Por ejemplo, vamos a demostrar que la concatenación es asociativa:

$$xs +\!\!+ (ys +\!\!+ zs) = (xs +\!\!+ ys) +\!\!+ zs$$

para cada lista finita xs, ys y zs.

Si bien aparecen tres listas en esta propiedad, la demostración se hace por inducción sobre una de ellas: xs. Llamaremos $P.xs$ a la propiedad a probar:

$$P.xs : xs +\!\!+ (ys +\!\!+ zs) = (xs +\!\!+ ys) +\!\!+ zs$$

Caso base:

$$\begin{array}{ll} & [\,] +\!\!+ (ys +\!\!+ zs) \\ = & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\ & ys +\!\!+ zs \\ = & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\ & ([\,] +\!\!+ ys) +\!\!+ zs \end{array}$$

Paso inductivo

Suponemos que $P.xs$ es cierta para cualquier lista xs con $\#xs = n$, entonces probamos $P.(x \triangleright xs)$:

$$
\begin{array}{ll}
 & (x \triangleright xs) +\!\!+ (ys +\!\!+ zs) \\
= & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\
 & x \triangleright (xs +\!\!+ (ys +\!\!+ zs)) \\
= & \quad \langle \text{ hipótesis inductiva, pues } \#xs = n \rangle \\
 & x \triangleright ((xs +\!\!+ ys) +\!\!+ zs) \\
= & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\
 & x \triangleright (xs +\!\!+ ys) +\!\!+ zs \\
= & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\
 & ((x \triangleright xs) +\!\!+ ys) +\!\!+ zs
\end{array}
$$

Otra de las propiedades que vimos relaciona la longitud con la concatenación:

$$\#(xs +\!\!+ ys) = \#xs + \#ys$$

para cualquier par de listas finitas xs y ys.

Haremos la prueba por inducción sobre xs, con lo cual:

$$P.xs : \#(xs +\!\!+ ys) = \#xs + \#ys$$

Caso base:

$$
\begin{array}{ll}
 & \#([\,] +\!\!+ ys) \\
= & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\
 & \#ys \\
= & \quad \langle \text{ elemento neutro de la suma} \rangle \\
 & 0 + (\#ys) \\
= & \quad \langle \text{ definición recursiva de } \# \rangle \\
 & (\#[\,]) + (\#ys)
\end{array}
$$

Paso inductivo

Suponemos que $P.xs$ es cierta para cualquier lista xs con $\#(xs) = n$, entonces probamos $P.(x \triangleright xs)$:

$$
\begin{array}{ll}
 & \#((x \triangleright xs) +\!\!+ ys) \\
= & \quad \langle \text{ definición recursiva de } +\!\!+ \rangle \\
 & \#(x \triangleright (xs +\!\!+ ys)) \\
= & \quad \langle \text{ definición recursiva de } \# \rangle \\
 & 1 + (\#(xs +\!\!+ ys)) \\
= & \quad \langle \text{ hipótesis de inducción, pues } \#xs = n \rangle \\
 & 1 + (\#xs) + (\#ys) \\
= & \quad \langle \text{ asociatividad de la suma y definición recursiva de } \# \rangle \\
 & (\#(x \triangleright xs)) + (\#ys)
\end{array}
$$

11.6. Verificación de programas con listas

Utilizaremos inducción sobre listas también para la verificación de programas, es decir, probaremos que una función satisface cierta especificación por inducción.

Veamos un ejemplo, la función $map : (A \to B) \to [A] \to [B]$, la cual aplica una función a cada elemento de una lista (por ejemplo, $map.cuadrado.[3,5] = [9,25]$, o $map.par.[3,5] = [false, false]$), puede definirse recursivamente de la siguiente manera:

$$\begin{array}{lcl} map.f.[] & \doteq & [] \\ map.f.(x \triangleright xs) & \doteq & f.x \triangleright map.f.xs \end{array}$$

Probaremos que esta definición satisface la siguiente especificación:

$$\begin{array}{ll} pre: & true \\ post: & (\forall i : 0 \leq i < \#xs : (map.f.xs).i = f.(xs.i)) \end{array}$$

Primero derivamos la propiedad a probar, a partir de la especificación dada, esto corresponde a plantear:

$$P.xs : pre \Rightarrow post$$

puesto que $true \Rightarrow p \equiv p$, por ser *true* neutro a izquierda de la implicancia, nuestra propiedad a demostrar resulta:

$$P.xs : (\forall i : 0 \leq i < \#xs : (map.f.xs).i = f.(xs.i))$$

Ahora probamos que $P.xs$ es cierta para cualquier xs por aplicando la técnica de inducción sobre listas.

Caso base:

$$\begin{array}{ll} & (\forall i : 0 \leq i < \#[] : (map.f.[]).i = f.([].i) \\ = & \quad \langle \text{definición recursiva de } \# \rangle \\ & (\forall i : 0 \leq i < 0 : (map.f.[]).i = f.([].i) \\ = & \quad \langle \text{aritmética } 0 \leq i < 0 \equiv false \rangle \\ & (\forall i : false : (map.f.[]).i = f.([].i) \\ = & \quad \langle \text{Rango Vacío } \forall \rangle \\ & true \end{array}$$

Paso inductivo

Suponemos que $P.xs$ es cierta para cualquier lista xs con $\#(xs) = n$, y probamos $P.(x \triangleright xs)$, es decir,

$$\begin{array}{lll} \textbf{Hip}: & P.xs: & (\forall i : 0 \leq i < \#xs : (map.f.xs).i = f.(xs.i)) \\ \textbf{Tes}: & P.(x \triangleright xs): & (\forall i : 0 \leq i < \#(x \triangleright xs) : (map.f.(x \triangleright xs)).i = f.((x \triangleright xs).i)) \end{array}$$

Tomaremos toda la tesis demostraremos que la misma es equivalente a *true*.

$$
\begin{array}{ll}
 & (\forall i : 0 \leq i < \#(x \triangleright xs) : (map.f.(x \triangleright xs)).i = f.((x \triangleright xs).i) \\
= & \quad \langle \text{definición de } map\text{; definición de } \# \rangle \\
 & (\forall i : 0 \leq i < \#xs + 1 : (f.x \triangleright map.f.xs).i = f.((x \triangleright xs).i)) \\
= & \quad \langle \text{ Sep. de un Termino} \rangle \\
 & (f.x \triangleright map.f.xs).0 = f.((x \triangleright xs).0)) \wedge \\
 & (\forall i : 1 \leq i < \#xs + 1 : (f.x \triangleright map.f.xs).i = f.((x \triangleright xs).i)) \\
= & \quad \langle \text{ Def. de Indexar dos veces; Cambio de Var Dummy } i := i + 1 \rangle \\
 & f.x = f.x \wedge \\
 & (\forall i : 1 \leq i + 1 < \#xs + 1 : (f.x \triangleright map.f.xs).(i + 1) = f.((x \triangleright xs).(i + 1))) \\
= & \quad \langle \text{ Reflexividad } =\text{; Neutro } \wedge \text{ ; Indexación dos veces} \rangle \\
 & (\forall i : 1 \leq i + 1 < \#xs + 1 : (map.f.xs).i = f.(xs.i)) \\
= & \quad \langle \text{ Aritmética en el Rango: } 1 \leq i + 1 < \#xs + 1 \equiv 0 \leq i < \#xs \text{Leibniz en el Rango} \rangle \\
 & (\forall i : 0 \leq i < \#xs : (map.f.xs).i = f.(xs.i)) \\
= & \quad \langle \text{ Hip de Inducción } \rangle \\
 & true\,.
\end{array}
$$

Luego, teniendo las pruebas de que el caso base se cumple, y que el paso inductivo también, podemos afirmar que nuestra propiedad:

$$(\forall xs : [A] : \#xs \geq 0 : P.xs)$$

con,

$$P.xs : \quad (\forall i : 0 \leq i < \#xs : (map.f.xs).i = f.(xs.i))$$

es cierta. Por lo tanto, hemos verificado que nuestro programa *map*, satisface la especificación dada.

11.7. Ejercicios

Aclaración y Notación

El formato que seguiremos para las demostraciones por inducción de esta ejercitación deberá responder a la siguiente estructuración de la prueba. [1]

Queremos probar [1] $(\forall n : n \geq \text{✎ } n_0 : P.n)$ donde

$$P.n : \text{✎}E_n \quad [2]$$

Caso Base $n = \text{✎ } n_0$ [3]

$$\ll \textit{Demostración de que} P.n_0 \equiv true \gg$$

[1] El símbolo ✎ nos indica donde debemos instanciar cada caso particular de la propiedad.

Paso Inductivo [4]
Probaremos $P.(n+1)$ suponiendo que la propiedad se cumple para:

$$P.n_0, P.(n_0+1), \ldots, P.n \quad [4.1]$$

HI: $P.n_0 : \text{✎}E_{n_0}, P.(n_0+1) : \text{✎}\ E_{n_0+1}, \ldots, P.n : \text{✎}\ E_n$. [4.2]
TESIS: $P.(n+1) : \text{✎}\ E_{n+1}$

$$\ll \textit{Demostración de que } P.(n+1) \equiv \textit{true} \gg$$

donde primeramente enunciaremos la propiedad que demostraremos [1], luego describiremos la expresión sobre la cual aplicaremos el principio de inducción [2], dejando expuesta la variable que pertenece al conjunto inductivo (en este caso los naturales). Por último, separaremos los dos casos que conforman la prueba por inducción: el(los) caso(s) base [3] y el paso inductivo [4]. En este último paso, muchas veces no utilizaremos todas las hipótesis inductivas tal cual lo enunciamos en [4.1] y [4.2], por lo que comúnmente se suele escribir en los mismos sólo las hipótesis utilizada, como por ejemplo, la última hipótesis: $P.(n) : E_n$, o las últimas dos hipótesis: $P.(n-1) : E_{n-1}$, $P.(n) : E_n$ siempre dependiendo de la propiedad a demostrar. Estas dos situaciones son las más usuales, pero pueden existir otros problemas donde necesitemos utilizar un subconjunto de las hipótesis distintas de la última y la penúltima.

11.1 Probar las siguientes expresiones aritméticas por inducción sobre n

a) Para $n \geq 0$, $(\sum\ i : 1 \leq i \leq n : i) = n \times (n+1)/2$.

b) Para $n \geq 0$, $(\sum\ i : 0 \leq i < n : 2 \times i + 1) = n^2$.

c) Para $n \geq 0$, $(\sum\ i : 0 \leq i < n : 3^i) = (3^n - 1)/2$.

d) Para $n \geq 0$, $(\sum\ i : 0 \leq i \leq n : i^2) = n \times (n+1) \times (2 \times n + 1)/6$.

11.2 Probar por inducción sobre n que $4^n - 1$ es divisible por 3 cuando $n \geq 0$. Es decir,

$$(\forall n : n \geq 0 : (4^n - 1)\ \mathbf{mod}\ 3 = 0)\,.$$

Para la prueba le resultarán útiles las siguientes propiedades del operador **mod** con $n \neq 0$:

mod_+ $((a\ \mathbf{mod}\ n) + (b\ \mathbf{mod}\ n))\mathbf{mod}\ n = (a+b)\ \mathbf{mod}\ n$
mod_× $((a\ \mathbf{mod}\ n) \times (b\ \mathbf{mod}\ n))\mathbf{mod}\ n = (a \times b)\ \mathbf{mod}\ n$

11.3 Probar por inducción sobre n que $n^2 \leq 2^n$ cuando $n \geq 4$.

Observación: Resulta últil el teorema ya demostrado en el ejemplo 11.5 mediante el cual se afirma que:

$$(\forall\, n : 3 \leq n : 2 \times n + 1 < 2^n)\,.$$

11.4 Dada siguiente definición de la función de *fibonacci*:

$$\begin{array}{lcl} fib.0 & \doteq & 0 \\ fib.1 & \doteq & 1 \\ fib.(n+2) & \doteq & f.n + fib.(n+1) \end{array}$$

Probar las siguientes expresiones aritméticas por inducción sobre n,

a) Para $n \geq 0$, $(\sum\ i : 0 \leq i \leq n : fib.i) = fib.(n+2) - 1$.

b) Para $n \geq 1$, $(\sum\ i : 0 \leq i \leq n : (-1)^i \times fib.i) = (-1)^n \times fib.(n-1) - 1$.

c) Para $n \geq 0$, $(\sum\ i : 0 \leq i \leq n : fib.(2 \times i + 1)) = fib.(2 \times n + 2)$.

11.5 Definimos la función f recursivamente así:

$$\begin{array}{rcl} f.0 & \doteq & 0 \\ f.(n+1) & \doteq & 2 \times f.n + 1 \end{array}$$

Probar por inducción que $f.n = 2^n - 1$ cuando $n \geq 0$.

11.6 Demostrar cual es el error en el siguiente razonamiento, que demuestra que un grupo cualquiera de gatos está compuesto sólo por gatos negros.

> *La demostración es por inducción en el número de gatos de un grupo dado. Para el caso de* 0 *gatos el resultado es inmediato, dado que todo gato del grupo es negro. Consideremos ahora un grupo de* $n + 1$ *gatos. Tomemos un gato cualquiera del grupo. El grupo de gatos restantes tiene n gatos, y por hipótesis inductiva son todos negros. Luego, tenemos n gatos negros y uno que no sabemos aún de que color es. Intercambiamos ahora el gato que sacamos con uno del grupo y un grupo de n gatos. Por hipótesis inductiva los n gatos son negros, pero como el que tenemos aparte también es negro, entonces tenemos que los* $n + 1$ *gatos son negros.*

11.7 Dada siguiente especificación y definición de la función de *Npot*:

$$\begin{array}{l} Npot : \mathsf{Nat} \longrightarrow \mathsf{Real} \longrightarrow \mathsf{Real} \\ \text{Pre :} \quad x \neq 0 \\ \text{Post :} \quad Npot.n.x = x^n \end{array}$$

$$\begin{array}{lcl} Npot : \mathsf{Nat} \longrightarrow \mathsf{Real} \longrightarrow \mathsf{Real} & & \\ Npot.n.x & \doteq & (\quad x \neq 0 \wedge n = 0 \quad \rightarrow \quad 1 \\ & & \Box \quad x \neq 0 \wedge n > 0 \quad \rightarrow \quad x \times Npot.(n-1).x \\ & &) \end{array}$$

verifique que *Npot* satisface la especificación dada.

11.8 Dada siguiente especificación y definición de la función de *Nsum*:

$Nsum$: Nat $\longrightarrow$ Nat
Pre: true
Post: $Nsum.n = (\Sigma i \ : 1 \leq i \leq n : i)$

$$\begin{array}{lcl} Nsum : \mathsf{Nat} \longrightarrow \mathsf{Nat} & & \\ Nsum.0 & \doteq & 0 \\ Nsum.(n+1) & \doteq & (n+1) + Nsum.n \end{array}$$

verifique que $Nsum$ satisface la especificación dada.

11.9 Consideremos las definiciones de las funciones vistas en el Capítulo 8:

i) $hd.(x \triangleright xs) \doteq x$ ii) $tl.(x \triangleright xs) \doteq xs$

iii)
$$\begin{array}{lcl} init.[x] & \doteq & [\,] \\ init.(x \triangleright x' \triangleright xs) & \doteq & x \triangleright (init.(x' \triangleright xs)) \end{array}$$

iv)
$$\begin{array}{lcl} last.[x] & \doteq & x \\ last.(x \triangleright x' \triangleright xs) & \doteq & last.(x' \triangleright xs) \end{array}$$

Demostrar las siguientes propiedades:

a) $init.xs = xs \uparrow (\#xs - 1)$

b) $init.(xs +\!\!+ [x]) = xs$

c) $last.(xs +\!\!+ [x]) = x$

11.10 Probar las siguientes propiedades de las operaciones sobre listas. Considere en cada caso que $n \geq 0$ y $m \geq 0$.

a) $(xs \uparrow n) +\!\!+ (xs \downarrow n) = xs$

b) $(xs \downarrow n) \uparrow m = (xs \uparrow (m+n)) \downarrow n$

c) $(xs \downarrow n) \downarrow m = xs \downarrow (m+n)$

11.11 Probar las siguientes propiedades:

1. $(\forall\, i : \ 0 \leq i < \#xs \ : (xs +\!\!+ ys)[i] = xs[i])$
2. $(\forall\, i : \ \#xs \leq i < \#(xs +\!\!+ ys) \ : (xs +\!\!+ ys)[i] = ys[(i - \#xs)])$

11.12 Verifique si los siguientes programas satisfacen la especificación que los acompaña:

a)
$$\begin{array}{l} sum : [\mathsf{Int}] \longrightarrow \mathsf{Int} \\ sum.[\,] \doteq 0 \\ sum.(x \triangleright xs) \doteq x + sum.xs \end{array}$$

$$\begin{array}{l} sum : [\mathsf{Int}] \longrightarrow \mathsf{Int} \\ \textit{Pre: } true \\ \textit{Post: } sum.xs = (\Sigma\, i :\ 0 \leq i < \#xs :\ xs.i) \end{array}$$

b)
$$\begin{array}{l} prod : [\mathsf{Int}] \longrightarrow \mathsf{Int} \\ prod.[\,] \doteq 1 \\ prod.(x \triangleright xs) \doteq x \times prod.xs \end{array}$$

$$\begin{array}{l} prod : [\mathsf{Int}] \longrightarrow \mathsf{Int} \\ \textit{Pre: } true \\ \textit{Post: } prod.xs = (\Pi\, i :\ 0 \leq i < \#xs :\ xs.i) \end{array}$$

c)
$$\begin{array}{l} par : [\mathsf{Nat}] \longrightarrow \mathsf{Bool} \\ par.[\,] \doteq true \\ par.(x \triangleright xs) \doteq x \,\mathbf{mod}\, 2 = 0 \wedge par.xs \end{array}$$

$$\begin{array}{l} par : [\mathsf{Nat}] \longrightarrow \mathsf{Bool} \\ \textit{Pre: } true \\ \textit{Post: } par.xs = (\forall\, i :\ 0 \leq i < \#xs :\ xs.i \,\mathbf{mod}\, 2 = 0) \end{array}$$

d)
$$\begin{array}{l} nulo : [\mathsf{Int}] \longrightarrow \mathsf{Bool} \\ nulo.[\mathrm{x}] \doteq x = 0 \\ nulo.(x' \triangleright x \triangleright xs) \doteq x' = 0 \wedge nulo.(x \triangleright xs) \end{array}$$

$$\begin{array}{l} nulo : [\mathsf{Int}] \longrightarrow \mathsf{Int} \\ \textit{Pre: } \#xs > 0 \\ \textit{Post: } nulo.xs = (\forall\, i :\ 0 \leq i < \#xs :\ xs.i = 0) \end{array}$$

11.13 Probar que la siguiente definición de la función $split : \mathsf{Nat} \to [A] \to ([A], [A])$:

$$\begin{array}{lcl} split.0.xs & \doteq & ([\,], xs) \\ split.(n+1).[\,] & \doteq & ([\,], [\,]) \\ split.(n+1).(x \triangleright xs) & \doteq & (x \triangleright ys, zs) \\ & & [\![\, (ys, zs) \doteq split.n.xs \,]\!] \end{array}$$

satisface la especificación:

$$\begin{array}{ll} pre: & true \\ post: & split.n.xs = (xs \uparrow n, xs \downarrow n) \end{array}$$