Induction

Foundations for Programming Languages
MASTER IN SOFTWARE AND SYSTEMS
Universidad Politécnica de Madrid/IMDEA Software

October 13, 2017

To be turned in by October 31, 2017. Send them to me by mail to `jmarino@fi.upm.es`.

Inducción clásica sobre números naturales

Los siguientes ejercicios pueden resolverse usando la regla de demostración por inducción mostrada en la figura 1.[1]

Exercise 1. Demuestra que $\sum_{i=1}^{n} i = n(n+1)/2$.

Exercise 2. Demuestra que la suma de los n primeros números impares es n^2.

Exercise 3. Encuentra una fórmula polinómica para $\sum_{i=1}^{n} i^2$ y demuestra su corrección.

Exercise 4. Demuestra la terminación del algoritmo de Euclides para calcular el máximo común divisor (mcd) de dos números naturales:

```
int mcd(int a, int b) {
  int n = a;
  int m = b;
  while (n != m) {
    if (n > m) {
      n = n - m;
    } else {
      m = m - n;
    }
  }
  return n;
}
```

Exercise 5. Si consideramos el vértice de un *triángulo de Pascal* (aka de Tartaglia, de Khayyam, de Yang Hui, etc.) su fila 0, demuestra que la suma de los elementos de su fila i es 2^i.

Exercise 6. Demuestra que las expresiones regulares $a(ba)^*$ y $(ab)^*a$ representan el mismo lenguaje.

Exercise 7. La formula de Euler para grafos planares afirma que el número de vértices más el número de "caras" (o regiones) es igual al número de arcos más dos. Demuestra por inducción.

[1] Hemos seguido el criterio tradicional de hacer que los números naturales comiencen en 1. Evidentemente, el principio es el mismo si asumimos el criterio (común en informática) de que los naturales comiencen en 0. En los ejercicios podéis usar el criterio que mejor se adecue en cada caso.

$$\frac{P(1) \qquad \forall m \in \mathbb{N}.\, P(m) \implies P(m+1)}{\forall n \in \mathbb{N}.\, P(n)}$$

Figura 1: Principio *clásico* de inducción sobre números naturales.

$$\frac{\forall m \in \mathbb{N}. \forall i < m.\, P(i) \implies P(m)}{\forall n \in \mathbb{N}.\, P(n)}$$

Figura 2: Principio generalizado de inducción sobre números naturales.

Exercise 8. El *principio del casillero* (*pigeonhole principle* en inglés) afirma que si se reparten m objetos en n ($m > n$) cajas, habrá al menos una caja con más de un objeto. Demuestra por inducción.

Exercise 9. Considera la siguiente sintaxis abstracta para expresiones aritméticas con variables:

$$e ::= 0 \mid 1 \mid v \mid e + e \mid e * e$$

Se ha definido un sistema deductivo para razonar sobre desigualdades ($\leq$) de la siguiente manera:

$$e \leq e \quad \text{(REFL)} \qquad\qquad 0 \leq e \quad (0\leq)$$

$$\frac{e \leq e' \qquad e' \leq e''}{e \leq e''}\ \text{(TRANS)} \qquad \frac{e_1 \leq e_2 \qquad e_3 \leq e_4}{e_1 + e_3 \leq e_2 + e_4}\ (+\text{MON}) \qquad \frac{e_1 \leq e_2 \qquad e_3 \leq e_4}{e_1 * e_3 \leq e_2 * e_4}\ (*\text{MON})$$

Demuestra la corrección del sistema. Es decir, si se ha podido demostrar $e \leq e'$ entonces, para cualquier asignación de variables el valor de e debe ser menor que el de e'.

Inducción general sobre números naturales

Los siguientes ejercicios pueden resolverse usando el principio de inducción general sobre naturales, mostrado en la Fig. 2.

Exercise 10. Un árbol binario se define como una hoja o bien un nodo conectado con dos árboles binarios. Demuestra que para todo árbol binario, el número de hojas es igual al número de nodos + 1.

Exercise 11. Demuestra que todo natural admite una única descomposición en factores primos.

Exercise 12. Explica la relación entre el principio de inducción general y el algoritmo de ordenación de secuencias *quicksort*.

Inducción bien fundada

Exercise 13. Explica por qué las reglas de inducción de las Figs. 1 y 2 son casos particulares de la inducción bien fundada (Fig. 3).

Exercise 14. Una forma alternativa de decir que una relación $\prec$ sobre S es bien fundada es diciendo que todo subconjunto no vacío $T \subseteq S$ admite un elemento *minimal* respecto a $\prec$. Demuestra que ambas definiciones son equivalentes.

$$\frac{\forall a \in S. \forall b \prec a.\ P(b) \implies P(a)}{\forall x \in S.\ P(x)}$$

Figura 3: Principio de inducción *bien fundada*. Aquí, $\prec$ es una relación sobre S que no admite cadenas descendentes infinitas de la forma $a_1 \succ a_2 \succ a_3 \succ a_4 \succ \cdots$.

Exercise 15. Demuestra que si $\prec_1$ y $\prec_2$ son relaciones de orden total y bien fundadas definidas sobre S_1 y S_2 respectivamente, entonces el *orden lexicográfico* $\prec_l$ definido como

$$\begin{array}{rcl} (a,b) \prec_l (c,d) & \text{si} & a \prec_1 c \\ (a,b) \prec_l (a,d) & \text{si} & b \prec_2 d \end{array}$$

es un orden total bien fundado sobre $S_1 \times S_2$.

Exercise 16. A continuación definimos tres relaciones sobre cadenas de caracteres. Dos de ellas son bien fundadas y la otra no. Para las bien fundadas, justifica por qué lo son. Para la que no lo es, proporciona una cadena descendente infinita.

a. La relación $\prec_1$ es el orden lexicográfico basado en el alfabeto, es decir:

$$\begin{array}{rcl} \varepsilon \prec_1 s & \text{si} & s \neq \varepsilon \\ a_1 s \prec_1 b_1 t & \text{si} & a_1 \text{ precede a } b_1 \text{ en el alfabeto, o} \\ & & a_1 = b_1 \wedge s \prec_1 t \end{array}$$

b. La segunda relación ordena cadenas según su longitud:

$$a_1 a_2 \ldots a_j \prec_2 b_1 b_2 \ldots b_k \quad \text{si} \quad j < k$$

c. La tercera relación, $\prec_3$, combina las dos anteriores al usar $\prec_2$ con cadenas de distinta longitud y $\prec_1$ con cadenas de igual longitud:

$$a_1 a_2 \ldots a_j \prec_2 b_1 b_2 \ldots b_k \quad \text{si} \quad j < k \vee (j = k \wedge a_1 \ldots a_j \prec_1 b_1 \ldots b_k$$

Exercise 17. Define un tipo de datos Haskell *BTree* para los árboles binarios del ejercicio 10. Define funciones $nodes :: BTree \to Int$ y $leaves :: BTree \to Int$. Define una relación de orden estructural entre elementos del tipo de datos *BTree* y úsala para demostrar que $\forall t \in BTree.\ nodes(t) = leaves(t) + 1$.

Exercise 18. Demuestra la terminación de la función de Ackermann, definida así:

```
ackermann :: (Int, Int) -> Int
ackermann (m, n) = if (m <= 0)
                   then n + 1
                   else if n <= 0
                        then ackermann (m - 1, 1)
                        else ackermann (m - 1, ackermann (m, n-1))
```

Exercise 19. Considera el lenguaje y sistema deductivo del ejercicio 9. Demuestra que si se ha conseguido demostrar $e \leq e'$ (sin o con la ayuda de supuestos previos) entonces existe una demostración de $e \leq e'$ que no hace uso de la regla (TRANS) inmediatamente después de dos usos de la regla (+MON). Es decir, se puede eliminar el siguiente patrón de cualquier demostración:

$$\dfrac{\dfrac{x \leq y \qquad x' \leq y'}{x + x' \leq y + y'}\,{+}\text{MON} \qquad \dfrac{y \leq z \qquad y' \leq z'}{y + y' \leq z + z'}\,{+}\text{MON}}{x + x' \leq z + z'}\,\text{TRANS}$$