Introducción a los Algoritmos

Ejercicios seleccionados

Pedro Sánchez Terraf

1. Utilizá las definiciones intuitivas de los operadores de listas para evaluar las siguientes expresiones. Subrayá la subexpresión resuelta en cada paso justificado. Luego usá un intérprete de `haskell` para verificar los resultados. [*Práctico 2*]

   a) $\#[5, 6, 7]$
   b) $[5, 6, 7] \uparrow 2$

2. Definir la función $mayor3 : (Int, Int, Int) \to (Bool, Bool, Bool)$, que dada una una terna de enteros devuelve una terna de valores booleanos que indica si cada uno de los enteros es mayor que 3.

   Por ejemplo: $mayor3.(1, 4, 3) = (False, True, False)$ y $mayor3.(5, 1984, 6) = (True, True, True)$

3. Una función de **map** es aquella que dada una lista devuelve otra lista cuyos elementos son los que se obtienen de aplicar una función a cada elemento de la primera en el mismo orden y con las mismas repeticiones (si las hubiere). Por ejemplo: $duplica : [Int] \to [Int]$ devuelve cada elemento de la lista multiplicado por 2.

   Definí recursivamente las siguientes funciones de map. [*Práctico 2*]

   a) $multiplica : Int \to [Int] \to [Int]$, que dado un número n y una lista, multiplica cada uno de los elementos por n.
   Por ejemplo: $multiplica.3.[3, 0, -2] = [9, 0, -6]$

4. Una función de **filter** es aquella que dada una lista devuelve otra lista cuyos elementos son los elementos de la primera que cumplan una determinada condición, en el mismo orden y con las mismas repeticiones (si las hubiere). Por ejemplo: $soloPares : [Int] \to [Int]$ devuelve aquellos elementos de la lista que son pares.

   Definí recursivamente las siguientes funciones filter. [*Práctico 2*]

   a) $mayoresQue10 : [Int] \to [Int]$, que dada una lista de enteros xs devuelve una lista sólo con los números mayores que 10 contenidos en xs,
   Por ejemplo: $mayoresQue10.[3, 0, -2, 12] = [12]$

5. Demostrá por inducción las siguientes propiedades. **Ayuda:** Recordá la definición de cada uno de los operadores implicados en cada expresión. [*Práctico 2*]

   a) $xs +\!\!+ [\,] = xs$ (la lista vacía es el elemento neutro de la concatenación)
   b) $xs +\!\!+ (ys +\!\!+ zs) = (xs +\!\!+ ys) +\!\!+ zs$ (la concatenación es asociativa)

6. Considerando la función $repetir : Nat \to Num \to [Num]$, que construye una lista de un mismo número repetido cierta cantidad de veces, definida recursivamente como:

$$\begin{aligned} repetir.0.x &\doteq [\,] \\ repetir.(n+1).x &\doteq x \triangleright repetir.n.x \end{aligned}$$

   demostrá que $\#repetir.n.x = n$. [*Práctico 2*]

7. Considerando las funciones *sum* y *duplica*, demostrá que: [*Práctico 2*]

$$sum.(duplica.xs) = 2 * sum.xs$$

8. ¿Están bien escritas las siguientes expresiones? Para evitar errores, introducí paréntesis de acuerdo a las reglas de precedencia, y en caso de ser posible escribí una tabla declarando el tipo de cada variable. [*Práctico 3*]

   a) $2 = 3 \vee 3 = 4 \vee a * a + 2 \leqslant b + 7$.

   b) $a \vee b = 3 + y$.

   c) $a + 2 \geqslant c \Rightarrow 3 + 2 < b \ \equiv \ c \equiv b = 2 * a$.

9. Evaluá las siguientes expresiones *booleanas*, subrayando la subexpresión resuelta en cada paso, y justificado [*Práctico 3*]. Por ejemplo:

$$
\begin{aligned}
& \underline{(\mathit{False} \vee \mathit{True})} \wedge \mathit{False} \equiv \mathit{False} \\
\equiv & \ \{ \text{ def. de } \vee \ \} \\
& \underline{\mathit{True} \wedge \mathit{False}} \equiv \mathit{False} \\
\equiv & \ \{ \text{ def. de } \wedge \ \} \\
& \mathit{False} \equiv \mathit{False} \\
\equiv & \ \{ \text{ def. de } \equiv \ \} \\
& \mathit{True}
\end{aligned}
$$

   a) $5 > 3 \wedge 3 > 1 \Rightarrow 5 > 1$

   b) $\mathit{False} \Rightarrow 2 + 2 = 5$

En `haskell` los distintos operadores booleanos se pueden escribir así:

$\neg p$	`not p`
$p \wedge q$	`p && q`
$p \vee q$	`p || q`
$p \equiv q$	`p == q`
$p \not\equiv q$	`p /= q`

10. Definir en `haskell` una función $\mathit{impl} : \mathit{Bool} \to \mathit{Bool} \to \mathit{Bool}$ que se comporte de acuerdo a la tabla de verdad de $\Rightarrow$. [*Práctico 3*]

11. Evaluar las siguientes fórmulas proposicionales, utilizando la siguiente asignación $p \doteq \mathit{True}$, $q \doteq \mathit{False}$ y $r \doteq \mathit{False}$. Subrayar la subexpresión resuelta en cada paso y justificar. [*Práctico 3*]

   a) $p \Rightarrow (q \equiv r)$.

12. Usá un intéprete de `haskell` para verificar los resultados anteriores. [*Práctico 3*]

13. Decidí si cada una de las siguientes fórmulas proposicionales son válidas o no. En caso que una fórmula no sea válida, decidí si es satisfactible o no. En todos los casos justificá con una tabla de verdad, un ejemplo o un contraejemplo, según corresponda. [*Práctico 3*]

   a) p

   b) $p \equiv p$

   c) $p \equiv p \equiv p$

14. Representá con una fórmula de lógica proposicional los siguientes enunciados. Usá el símbolo de predicado p para representar $(a < b)$, usá q para representar $(b < c)$, y r para $(a < c)$. [*Práctico 3*]

   a) $a < b < c$

   b) $(a \geq b$ y $b < c)$ o $(a \geq c)$

15. Escribí una fórmula proposicional para cada una de las siguientes frases, utilizando una variable proposicional para cada sentencia atómica, aclarando siempre el significado escogido para cada variable. [*Práctico 3*]

   a) Hoy es viernes, pero tengo clases de Algoritmos.

   b) Yo no voy de vacaciones y Juan y Pedro tampoco.

16. Formalizá los siguientes razonamientos en la lógica proposicional. [*Práctico 3*]

   a) Si hago mucho deporte estoy cansado. Estoy cansado, por lo tanto hago mucho deporte.

   b) Si mañana es martes, tenemos clase. Si mañana es martes y llueve, tenemos clase y llueve.

17. Demostrá que las siguientes fórmulas son teoremas justificando cada paso con el axioma o teorema aplicado. **Aclaración:** Desde ahora en adelante, en cada ejercicio se pueden utilizar los teoremas ya demostrados en los ejercicios anteriores. [*Práctico 4*]

    a) *Elemento absorbente de la conjunción*: $p \wedge \mathit{False} \equiv \mathit{False}$.
    b) *Elemento neutro de la conjunción*: $p \wedge \mathit{True} \equiv p$.
    c) *Modus Ponens*: $(p \Rightarrow q) \wedge p \equiv p \wedge q$.
    d) *Negación de una implicación*: $\neg(p \Rightarrow q) \equiv p \wedge \neg q$.

18. Decidí si cada una de las siguientes fórmulas proposicionales son válidas o no. En todos los casos justificá con una demostración o un contraejemplo, según corresponda.

    a) $p \wedge (q \equiv r) \equiv (p \wedge q) \equiv (p \wedge r)$
    b) $p \wedge (q \equiv r) \equiv (p \wedge q) \equiv (p \wedge r) \equiv p$

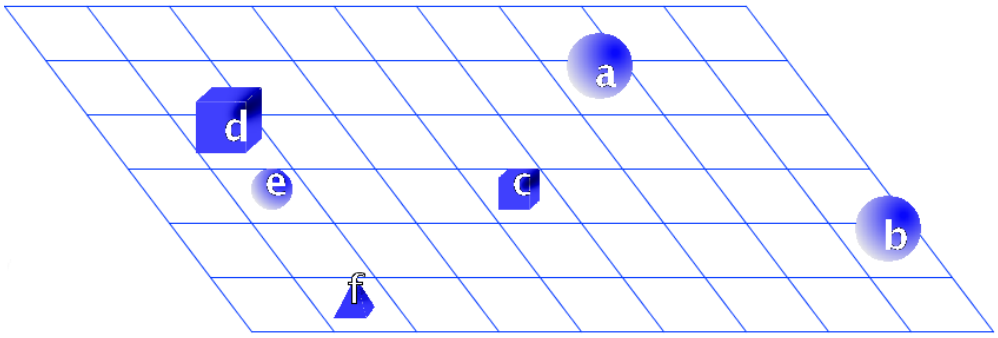
19. Construí un *único* modelo como los de `SAT`, en el que se satisfagan, simultáneamente, las siguientes propiedades:

    a) $\langle \forall x : : \mathit{rojo}.x \equiv \mathit{grande}.x \rangle$
    b) $\langle \exists x : : \langle \exists y : \neg(x = y) : \mathit{grande}.x \equiv \neg \mathit{rojo}.y \rangle \rangle$
    c) $\langle \forall x : \neg \mathit{grande}.x : \langle \exists y : : \mathit{esfera}.y \wedge \mathit{rojo}.y \rangle \rangle$
    d) $\langle \forall y : : (\neg \mathit{rojo}.y) \equiv \mathit{pirámide}.y \rangle$

20. Expresá el significado de cada una de las siguientes fórmulas en lenguaje natural. [*Práctico 5*]

    a) $\langle \forall x : x \in \mathit{Num} : \langle \exists y : y \in \mathit{Int} : x < y \rangle \rangle$
    b) $\langle \exists x : x \in \mathit{Num} : \langle \forall y : y \in \mathit{Int} : x < y \rangle \rangle$ ¿Es lo mismo que el anterior?

21. Dado el siguiente mundo



    i) Decidí si las siguientes sentencias son satisfechas o no (y por qué).
    ii) Expresá formalmente cada sentencia y verificala en `SAT`. [*Práctico 5*]

    a) `a` está a la derecha de todo.
    b) Nada está a la derecha de `b`.

22. Formalizá las siguientes sentencias escritas en lenguaje natural. [*Práctico 5*]

    a) El producto de dos impares es impar.
    b) x está en la lista xs.
    c) La lista xs consiste de 0's y 1's.
    d) Todos los elementos de xs son iguales.
    e) El primer elemento de xs es el máximo.

23. **Ejercicios de Cálculo de Predicados** (Práctico 5):