Puzzles, Juegos Matemáticos y Programación Funcional

Blas Ruiz

Dpto. de Lenguajes y Ciencias de la Computación. Universidad de Málaga.

www.lcc.uma.es\~blas

La matemática recreativa ($\mathcal{M}at\mathcal{R}ec$) (*recreational mathematics*) es una disciplina que incluye juegos matemáticos:

lógicos, puzzles, acertijos, problemas de ingenio, ...

El *Journal of Recreational Mathematics* es la publicación más seria sobre esta disciplina. (http://www.ashbacher.com/jrecmath.stm)

Tradicionalmente la $\mathcal{M}at\mathcal{R}ec$ ha sido utilizada para motivar:

✓ la lógica, aritmética, geometría ...elementales.

Hoy añadimos a la lista anterior: la enseñanza de la programación.

Objetivos

1. Motivar los principios de la Programación Funcional ($\mathcal{P}rog\mathcal{F}un$) vía ejemplos lúdicos extraídos de la $\mathcal{M}at\mathcal{R}ec$.
2. Ilustrar las características esenciales de un lenguaje funcional moderno como *Haskell*:

   • tipificación fuerte • definiciones con patrones
   • estructuras de datos . . . • ¡programas elegantes!

3. Justificar las construcciones esenciales de la $\mathcal{P}rog\mathcal{F}un$:

   • composición de funciones • listas por comprensión . . .

Principio

Elección de problemas introductorios (puzzles) que sean

• elementales • no triviales
• resolubles usando lo esencial de la $\mathcal{P}rog\mathcal{F}un$

Matemáticos con aportaciones importantes en $\mathcal{M}at\mathcal{R}ec$:

- Martin Gardner (1914-). Autor de 100 libros; célebre por su columna *Mathematical Games* en **Scientific American**. *Inspiración ¡ajá!* (1978) (Labor, 1981)
- Douglas Hofstadter (1945-). Su libro más conocido es *Gödel, Escher, Bach: an Eternal Golden Braid* (1979). Sucesor de Martin Gardner en la columna *Mathematical Games*.
- John Conway (1937-). Tiene aportaciones importantes en teoría de grupos, teoría de números, *combinatorial game theory*, *coding theory*. Inventor del Juego de la Vida y otros célebres puzzles, algunos aún no resueltos. *On Numbers and Games*, *Winning Ways for your Mathematical Plays*.

- Charles Dodgson (1832-1898) (Lewis Carroll). Profesor de matemáticas de la Universidad de Oxford cerca de 50 años; escribió dos obras maestras: *Alicia en el país de las maravillas* (1864), y *A través del espejo* (1871). *Symbolic Logic I* (1896), ... (http://www.guiascostarica.com)
- Raymond Smullyan (1919-). Lógico norteamericano. Apasionado por los problemas de ajedrez. Doctorado de Alonzo Church en Princeton (1959) (sistemas formales, teoría de la recursión, ...) Entre sus libros célebres:

  ¿Cómo se llama este libro? El enigma de Drácula y otros pasatiempos lógicos, Ed. Cátedra (1988). *What is the name of this book?* (1978)

  The Lady or the Tiger?

  Alice in puzzle-land (*Alicia en el País de la Adivinanzas*)

$$\mathcal{M}at\mathcal{R}ec \begin{cases} \text{Puzzles Lógicos} \begin{cases} \textbf{Puzzles con Diálogos} \\ \ldots \end{cases} \\ \ldots \end{cases}$$

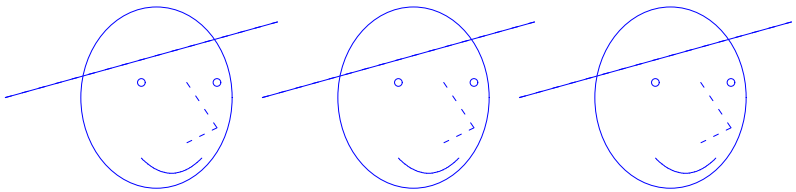
Puzzles con Diálogos

Deducir información (repartida por un árbitro a varios oradores) a partir de un diálogo cuyas frases contienen pistas sobre: (1) datos del problema, (2) identidad de los oradores, (3) información secreta de algunos oradores, ...

Puzzles con sombreros

Atribuido a Sebelik (1983), generalizado por J Conway, H Freudenthal, M Gardner, ...

En cierta habitación se encuentran Sebelik, Paco y Blas en fila india. *En otra contigua hay cinco sombreros (2 negros y 3 blancos). Se apaga la luz; un árbitro toma tres sombreros de la mesa y los coloca en las cabezas de las personas para después encender la luz.*



A continuación se escucha el siguiente diálogo:

Sebelik.— Yo no sé el color de mi sombrero.

Paco.— En ese caso, yo tampoco.

¿De qué color es el sombrero de Blas?

Puzzles con números en la frente (J. Conway)

John elige dos números $p, b \in [1..9]$ y un tercero $y \in [2..18]$

escribe p en la frente de *Paco* y b en la de *Blas*

escribe en una pizarra y junto a la suma $p + b$

$$10 = 5+?$$
$$9 = 5+?$$

Manuel desde otra habitación escucha el siguiente diálogo:

Blas .— *No sé el número de mi frente.*
Paco .— *Yo tampoco.*
Blas .— *Pues sigo igual.*
Paco .— *Y yo.*

¿Tiene sentido el diálogo?

Blas .— *¡Casi lo sé!*
Paco .— *Y yo.*
Blas .— *Me estoy poniendo nerviosillo.*
Paco .— *Yo también estoy de los nervios.*
Blas .— *¡Ajá, ya sé el mío!*

¿Sabe *Manuel* los números?

El Problema *P-S* (H Freudenthal)

Hans elige dos naturales distintos mayores que 1, entrega el producto a $\mathcal{PROD}$ y la suma a $\mathcal{SUM}$, que mantienen el siguiente diálogo:

$\mathcal{SUM}$.— *No se cómo vas a adivinar mi suma.*

$\mathcal{PROD}$.— *Entonces, no sé tu suma*

$\mathcal{SUM}$.— *Pues yo ya se tu producto.*

¿Cuáles eran los números elegidos por Hans?

✓ En el problema original (1969) la segunda frase es:

$\mathcal{PROD}$.— *¡Ajá!, entonces ya sé tu suma*

Información	*P-S*	*EnLaFrente*	*SigPrev*	*Sombreros*
global	$x, y > 1$ $x \neq y$	$p, b \in [1..9]$ $x, y \in [2..18]$	$p, b \in [1..9]$	2N, 3B
pública	Φ	pizarra (x, y)	Φ	Φ
privada	s, p	frente (x, y)	frente (x, y)	(p, b), b

En la isla de los Caballeros y los Pícaros

En esta isla habitan caballeros (sinceros) y pícaros (mentirosos). Ocasionalmente puede visitar la isla los espías (a veces mienten y otras no).

26 Me encuentro con A,B y C. Pregunto a A: ¿Eres Caballero o Pícaro? La respuesta de A es ininteligible, y pregunto a B: ¿Qué ha dicho A?

B.— A dice que es Pícaro.

C.— No creas a B, que está mintiendo.

¿Qué dijo A? ¿Qué son A y B?

40 Me encuentro con dos habitantes que discuten:

A.— No eres un Caballero.

B.— Tu sí que no lo eres.

Demuestra que uno de ellos es espía, y uno de ellos sincero.

En la isla de los Caballeros y los Pícaros (cont)

27 Pregunto a A: ¿cuantos caballeros hay entre vosotros? La respuesta de A es ininteligible, y pregunto a B: ¿Qué ha dicho?

B.— A dice que hay exactamente un caballero entre nosotros.

C.— B está mintiendo.

¿Qué dijo A? ¿Quiénes son A,B,C? Veo dos habitantes descansando bajo un árbol. Le pregunto a uno de ellos: ¿Es alguno de vosotros un caballero? Con la respuesta pude saber su personalidad.¿Qué eran?

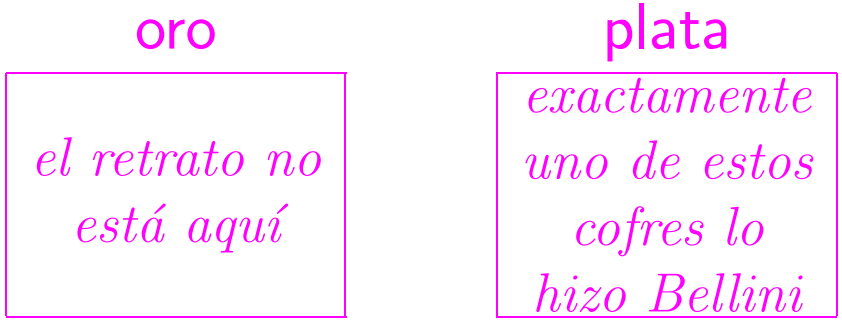
Los Cofres de Porcia

Porcia decide elegir esposo, no atendiendo a su belleza o bondad, sino a su agudeza mental o inteligencia.

Porcia tiene varios cofres (de oro, de plata, . . . algunos fabricados por Bellini, . . .). El pretendiente debe descubrir el cofre con su retrato de acuerdo con el *diálogo* dado en las inscripciones.

67a

69b



¿Vive el conde Drácula?

En Transilvania hay vampiros y humanos, y cada uno de ellos puede estar cuerdo o loco.

Los humanos son sinceros y los vampiros embusteros. Por lo contrario, los cuerdos creen que son ciertas las proposiciones ciertas, y falsas las falsas. Sin embargo, los locos creen que las proposiciones falsas son ciertas y las ciertas, falsas. (un humano loco se comporta *aparentemente* como un embustero, pero al contrario que éste, miente sin maldad).

167 Un transilvano dice: O soy humano o estoy cuerdo. ¿Es un vampiro?

170 Le preguntamos a T: ¿Eres un vampiro Loco? ¿Qué respondió si yo adiviné lo que era?

172 Otro dice: Estoy Loco. ¿Es un vampiro?

176 RS entabla una conversación con dos transilvanos:

RS, dirigiéndose a A.— ¿Es B humano?

A.— Eso creo.

RS, dirigiéndose a B.— ¿A es humano?

¿Qué respondió B?

178 RS, dirigiéndose a A: ¿Crees que eres formal? ¿Qué puedo determinar dependiendo de la respuesta?

¿Es posible deducir si Drácula vive a partir de frases como:

179 Si soy humano entonces Drácula vive aún.

180 Si estoy cuerdo entonces Drácula vive.

182 Si soy formal entonces Drácula vive.

190 ¿Qué preguntaremos a un transilvano para que, independientemente de su comportamineto y su respuesta, podamos saber si Drácula vive?

Clasificación de Puzzles

1. Con frases dependientes y oradores $infinitamente$ inteligentes pero sinceros (cada frase depende de la anterior):
   a) Sombreros (J Conway, H Freudenthal)
   b) Números en la frente (J Conway, M Gardner)

   Problema: modelar el comportamiento inteligente
2. Con frases independientes, pero posiblemente falsas:
   a) Problemas de $Alicia$: En la isla, El bosque del Olvido, ...
   b) La colección de Smullyan: Drácula, Cofres de Porcia, ...

   Problema: modelar la interpretación de las frases
3. Mezcla de los anteriores.

La Programación Funcional es una herramienta:

- sencilla
- suficiente
- atractiva

Tareas a realizar

1. – *Estudio del Problema*
2. – *Modelado de la Solucion en* Haskell
3. – *Conjeturas sobre la solucion*
4. – *Pruebas de éstas*
5. – *Variantes del Problema*

Programación Funcional

programar $\boxed{=}$ combinar funciones

✓ $\boxed{=}$ está sobrecargado

- Moses Schönfinkel (Soc. Mat. de Göttingen, 1920),
  Uber die Bausteine der mathematischen logik (1924),
  On the building blocks of mathematical logic

representar fórmulas a través de combinadores

$$\mathbf{S}\,x\,y\,z = x\,z\,(y\,z) \qquad \mathbf{K}\,x\,y = x \quad (Konstanzfunktion)$$

introduce la notación parcializada (*currying*)

- La Lógica Combinatoria de Haskell Curry (1930)
- Cálculo con funciones anónimas (B Russel?):

$$2\widehat{x} + y \rightsquigarrow \widehat{x}.2x + y \rightsquigarrow {}_{\wedge}x.2x + y \rightsquigarrow \lambda x.2x + y$$

- Alonzo Church,

  A set of postulates for the foundations of logic (1932)

Introduce el *Calculi of lambda conversion* o λ–cálculo:

$x\ y$	— variables
$\lambda\, x.\, E$	— abstracciones o funciones anónimas
$e\ h$	— aplicaciones

⊕ paréntesis y convenios para agilizar la notación:

$$\lambda x.\lambda y.x \qquad \lambda xy.x \qquad (\lambda xy.x)(\lambda z.(\lambda m.m)(\lambda m.m))(\lambda xy.y)$$

⊕ la aplicación a varios argumentos es implícita: $f\, a\, b\, c$

⊕ definiciones o ecuaciones para simplificar expresiones:

$$Kxy = x \qquad I = \lambda x.x \qquad F\, x\, y = y$$

$$K(\lambda z.I\, I)F$$

⊕ (beta) conversión un paso de evaluación: (redex)

$$\frac{}{(\lambda x.C)\ E \rightsquigarrow\ [x := E]C} \qquad \frac{E \rightsquigarrow E'}{\lambda x.E \rightsquigarrow \lambda x.E'} \qquad \frac{E \rightsquigarrow E'}{E\,M \rightsquigarrow E'\,M} \cdots$$

Ejemplo:

$$K(\lambda z.II)F \rightsquigarrow K(\lambda z.I)F \qquad K(\lambda z.II)F \rightsquigarrow (\lambda z.II)$$

⊕ evaluación en cero o más pasos

$$\frac{}{E \rightsquigarrow\!\!\!\!\rightarrow E} \qquad \frac{E \rightsquigarrow\!\!\!\!\rightarrow E' \quad E' \rightsquigarrow\!\!\!\!\rightarrow E''}{E \rightsquigarrow\!\!\!\!\rightarrow E''}$$

⊕ cómputo: uno o varios pasos.

$$K(\lambda z.II)F \rightsquigarrow\!\!\!\rightarrow \lambda z.I \qquad\qquad K(\lambda z.II)F \rightsquigarrow\!\!\!\rightarrow K(\lambda z.I)F$$

⊕ fin de cómputo: no quedan redexes (forma normal)

⊕ intérprete: estrategia de reducción

Teoremas de Church-Rosser:

1. $\rightsquigarrow\!\!\!\rightarrow$ es confluente

   Si $e \rightsquigarrow\!\!\!\rightarrow e' \wedge e \rightsquigarrow\!\!\!\rightarrow e''$ entonces, $\exists f | e' \rightsquigarrow\!\!\!\rightarrow f \wedge e'' \rightsquigarrow\!\!\!\rightarrow f$

2. La FN es única.
3. El cómputo final es independiente del evaluador.

En el λC podemos modelar la lógica, los números, las listas . . .

Usaremos definiciones ecuacionales

$$True = \lambda xy.x \qquad False = \lambda xy.y \qquad Cond = \lambda xyz.x\,y\,z$$

$$\& = \lambda xy.x\,y\,F \qquad not = \lambda x.x\,F\,T$$

$$Cond\ True\ e\ f \twoheadrightarrow e$$

Sea $\boxed{=}$ la igualdad sintáctica (que absorbe a la $=$ definicional):

$$Not\ T \boxed{=} (\lambda x.x\,F\,T)T \rightsquigarrow T\,F\,T \rightsquigarrow F$$

$$\lceil 0 \rceil = \lambda fx.x \qquad \lceil n \rceil = \lambda fx.f^{n+1}(x)$$

$$[\,] = \lambda xyz.y \qquad (:) = \lambda abf.f\ a\ b$$

$$head = \lambda x.x\ T \qquad tail = \lambda x.x\ F$$

$$[A] \boxed{=} (:) A [\,] \rightsquigarrow \lambda f.f\ A\ [\,]$$

Pero la igualdad $\boxed{=}$ es sustitutiva

$$head\ [A] \boxed{=} (\lambda x.x\ T)[A] \rightsquigarrow [A]T \boxed{=} (:)A[\,]T \twoheadrightsquigarrow T\ A[\,] \rightsquigarrow A$$

$$tail\ [A] \boxed{=} (\lambda x.x\ F)[A] \rightsquigarrow [A]F \boxed{=} (:)A[\,]F \twoheadrightsquigarrow F\ A[\,] \rightsquigarrow [\,]$$

Luego

$$head\ [A] \twoheadrightsquigarrow A \qquad tail\ [A] \twoheadrightsquigarrow [\,]$$

La programación funcional consiste en diseñar combinadores

⊕ Funciones Recursivas

Si $Y = \lambda f.(\lambda x.f(x\,x))\,(\lambda x.f(x\,x))$, entonces $Y\ f \twoheadrightarrow f\ (Y\ f)$, e.d.:

Si definimos una igualdad $\leftrightharpoons$ que incluya a $\boxed{=} \cup \twoheadrightarrow$ entonces la ecuación $M \leftrightharpoons f\ M$ tiene solución.

⊕ λ–definibilidad: base de la programación funcional

Una función $\varphi : \mathbb{N}^p \to \mathbb{N}$ se dice λ–definible si existe un término F tal que

$$\lceil \varphi(n_1, \ldots, n_p) \rceil \twoheadrightarrow F\ \lceil n_1 \rceil\ \ldots\ \lceil n_p \rceil$$

Teorema de Kleene: f es recursiva sii es λ–definible.

¿Qué lenguajes funcionales son buenos?

La lista es larga:

APL FP	simbólicos
LAMBDA-CÁLCULO LISP SCHEME	semi-puros
HOPE MIRANDA ORWELL HASKELL	algebraicos (tipos inductivos)

✓ característica esencial:	*tipificación* (clasificación)
✓ otras características:	*orden-superior y parcialización, polimorfismo general y restringido, evaluación no estricta,...*

El lenguaje Haskell (Curry)

Una agradable introducción a HASKELL ...

En nuestras páginas hay información adicional:

www.lcc.uma.es\~pepeg www.lcc.uma.es\razonandoConHaskell

- Surge (los 80) para estandarizar las propuestas académicas.
- Es un LAMBDA-CÁLCULO enriquecido:

$x,\ y$	— variables
$\lambda\, x\ \rightarrow\ E, \quad \lambda\, x\, y\ \rightarrow\ F$	— funciones anónimas
$e\ h$	— aplicaciones
$1, -8, +, \ldots \quad True, False,$	— aritmética, lógica
$[1,2], \quad 'a' : "juan", \quad ('a', 3.23), \quad length$	— funciones básicas

- Declaraciones previas de una amplia colección tipos:

$$\begin{array}{ll}
\textbf{data}\ Bool\ =\ True \mid False & \\
\textbf{data}\ Integer\ =\ ... \mid -1 \mid 0 \mid 1 \mid 2 \mid ... & \text{— enteros no acotados} \\
\textbf{data}\ Int\ =\ -2147483648 \mid ... \mid 2147483647 & \text{— [minBound..maxBound]} \\
\textbf{data}\ [a]\ =\ [\,] \mid a\ :\ [a] & \text{— listas}
\end{array}$$

- Organización en módulos ($import, hidding$) PRELUDE LIST
- Patrones en definiciones, orden superior:

$$\begin{array}{lll}
map & :: (a \to b) \to [a] \to [b] & \text{— polimorfismo+OS} \\
map\ f\ [\,] & = [\,] & \\
map\ f\ (x : xs) & = f\ x\ :\ map\ f\ xs & \text{— recursión}
\end{array}$$

$$aMayúsculas\ ::\ [Char] \to [Char]$$

$$aMayúsculas\ =\ map\ Char.toUpper \quad \text{— parcialización, cualificación}$$

- Operadores simbólicos, precedencia y asociatividad:

$$\begin{array}{l}\textbf{infixl}\ 1\ ==> \quad \text{— implicación lógica}\\ False\ ==>\ \_\ =\ True \quad \text{— patrón anónimo}\\ True\ ==>\ q\ =\ q\end{array}$$

- Ecuación con guardas, promoción numérica:

$$\begin{array}{l}fac\ n \mid n == 0\ =\ 1\\ \qquad\quad \mid n > 0\ \ =\ n\ *\ fac(n-1)\end{array}$$

$Main >\ fac\ 100$ — El sistema infiere el tipo de fac

$$\begin{array}{l}933262154439441526816992388562667004907159682643816214685929\\ 638952175999932299156089414639761565182862536979208272237582\\ 511852109168640000000000000000000000\ ::\ Integer\end{array}$$

$Main >\ length\ \$\ show\ \$\ fac\ 100$ — $\textbf{infixr}\ 0\ \$;\ f\ \$\ a\ =\ f\ a$

$158\ ::\ Int$

```
Prelude > map (/3) [1..3]    — promoción numérica
[0.333333333333333,0.666666666666667,1.0] :: Double
Prelude > filter (/= 's') "comes filetes"   — llamada parcial
"come filete" :: String
```

¿Con cuántos ceros termina el factorial de 100?

```
Main > ceros 100
  where ceros = length . takeWhile (=='0') . reverse . show . fac
24 :: Int    — ceros es una función local
```

- Tipos algebraicos (inductivos, dots)

```
data Natural = O | Suc Natural    — inductivo
masDos = Suc . Suc    — masDos :: Natural → Natural
data Pila a = P a (Pila a)    — recursivo, no inductivo
p = p    — p :: Pila a
q = P 1 q    — q :: Pila Int
```

- Evaluación no estricta

$$\textit{Prelude} > \textit{take}\ 10\ \textit{unos}\ \textbf{where}\ \textit{unos} = 1 : \textit{unos}$$
$$[1,1,1,1,1,1,1,1,1,1] :: \textit{Integer}$$

- Clases de Tipos:

$$\textbf{class}\ \textit{Eq}\ a\ \textbf{where}$$
$$(==), (/=) :: a \to a \to \textit{Bool}$$
$$\text{— miembros por defecto}$$
$$x == y = not(x \neq y)$$
$$x \neq y = not(x == y)$$

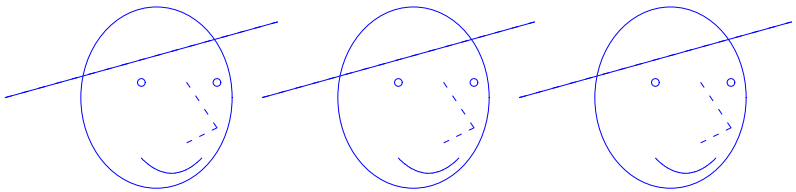
$$\textbf{instance}\ \textit{Eq}\ \textit{Natural}\ \textbf{where}$$
$$O == O = \textit{True}$$
$$\textit{Suc}\ x == \textit{Suc}\ y = x == y$$

- Un intérprete/compilador libre muy *cuidado* www.haskell.org

Puzzles con sombreros

Atribuido a Sebelik (1983), generalizado por J Conway, H Freudenthal, M Gardner, ...

En cierta habitación se encuentran Sebelik, Paco y Blas en fila india. *En otra contigua hay cinco sombreros (2 negros y 3 blancos). Se apaga la luz; un árbitro toma tres sombreros de la mesa y los coloca en las cabezas de las personas para después encender la luz.*



A continuación se escucha el siguiente diálogo:

Sebelik.— Yo no sé el color de mi sombrero.

Paco.— En ese caso, yo tampoco.

¿De qué color es el sombrero de Blas?

Sombreros y diálogo en HASKELL

La representación de colores es simple:

$$\textbf{data}\ Color\ =\ B \mid N\ \textbf{deriving}\ (Eq, Show)$$

$Sombreros > B == N$
$False$
$Sombreros > B\ \neq N$
$True$

Necesitamos un predicado inteligente para cada frase:

$$\textbf{type}\ Frase\ a\ =\ a\ \rightarrow\ Bool \quad \text{— POLIMORFISMO}$$

$$\begin{array}{lll} sebelik & :: & Frase\ (Color, Color) \\ paco & :: & Frase\ Color \\ blas & :: & [Color] \quad \text{— la lista de posibles sombreros} \end{array}$$

$sebelik$ actúa sobre pares de colores (los sombreros que ve).

Algunas funciones sencillas usando "Listas por Comprensión":

$$cubo :: [a] \to [(a,a,a)] \quad \text{— Polimorfismo}$$

$$cubo\ xs = [\,(x,y,z) \mid x \leftarrow xs,\ y \leftarrow xs,\ z \leftarrow xs]$$

— una lista por comprensión captura "conjuntos por comprensión":

— $\mathbb{R}^3 = \{(x,y,z) | x \in \mathbb{R}, y \in \mathbb{R}, z \in \mathbb{R}\}$

$$elegibles :: [(Color, Color, Color)]$$

$$elegibles = [\,t \mid \underbrace{t \leftarrow cubo\,[B,N]}_{generador}, \underbrace{t \neq (N,N,N)}_{guarda}]$$

— hay suficientes blancos

$Sombreros > cubo\,[B,N]$

$[(B,B,B),(B,B,N),(B,N,B),(B,N,N),$
$(N,B,B),(N,B,N),(N,N,B),(N,N,N)]$

$Sombreros > elegibles$

$[(B,B,B),(B,B,N),(B,N,B),(B,N,N),$
$(N,B,B),(N,B,N),(N,N,B)]$

¿Cómo es de inteligente Sebelik?

$$sebelik :: Frase\ (Color, Color)$$

$$sebelik\ (p, b) = masDeUno\ [\ s \mid \underbrace{(s, p', b') \leftarrow elegibles}_{generador}, \underbrace{(p', b') == (p, b)}_{guarda}]$$

```
Sombreros > sebelik (B, N)    — ¿Sebelik puede ver uno Blanco y otro Neg
True                          — Sí
Sombreros > sebelik (N, N)
False
```

¿Y el comportamiento inteligente de Paco?

$$paco\ b\ =\ masDeUno\ [\ p\ |\ p\ \leftarrow\ [B, N], sebelik\ (p, b)\ ]$$

— imagino un color p para mi sombrero
— y razono como Sebelik

```
Sombreros > paco B    — ¿Paco puede ver un sombrero Blanco?
True                  — Sí
Sombreros > paco N
False
```

La lista $blas$ calcula los posibles colores del sombrero de Blas:

$$blas\ =\ [\ b\ |\ b\ \leftarrow\ [B, N],\ paco\ b\ ]$$

```
Sombreros > blas
[B] :: [Color]    — esto es una demostración
```

Veamos ahora la función $masDeUno$:

```
masDeUno :: Eq a => [a] → Bool    — polimorfismo restringido
masDeUno = varios . quitaRep

   — el punto '.' representa la composición de funciones
(f . g) x = f (g x)

quitaRep :: Eq a => [a] → [a]
quitaRep (x : xs) = x : quitaRep [ y | y ← xs, y ≠ x]
quitaRep []       = []    — recursión

data [a] = [] | a : [a]
```

```
Sombreros > quitaRep [3, 1, 2, 1, 3, 1, 4]
[3, 1, 2, 4]
Sombreros > quitaRep "esta frase es muy larga"
"esta frmuylg"
```

$$varios :: [a] \to Bool \quad \text{— polimorfismo}$$
$$varios\ (\_ : \_ : \_) = True \quad \text{— el símbolo \_ es el patrón anónimo}$$
$$varios\ \_ = False$$

— Otra alternativa vía la función *length*

$$varios\ xs = length\ xs \geq 2$$

$$length :: [a] \to Integer \quad \text{— polimorfismo irrestringido}$$

$$length\ (\_ : xs) = 1 + length\ xs$$
$$length\ [\,] = 0$$

— Otra forma directa ¿más elegante?

$$varios = (\geq 2)\ .\ length$$

Mas puzzles con sombreros

Martin Gardner (en *Inspiración ¡Ajá!*) considera dos variantes (H Freudenthal y J Conway)

... n personas colocadas en *fila india*. En la mesa hay $n-1$ sombreros negros y n blancos ... un *árbitro* elige n sombreros ... Sucesivamente, de atrás hacia adelante, todas (salvo la última) dicen lo mismo: ¡No sé el color de mi sombrero!. ¿De qué color es el sombrero de la primera persona de la fila?

¿Qué ocurre si las n personas pueden moverse libremente, de forma que cada una ve las cabezas de las restantes?

Demostrad que alguna de ellas afirmará en algún momento que conoce el color de su sombrero.

Generalización (4 personas, 5 ...)

Generalizamos el programa anterior **EJERCICIO**

✓ Lo anterior proporciona un razonamiento ¡ajá!

Sea la fila india: (P_1 habla el 1º)

$$P_1 P_2 P_3 \ldots P_n$$

De la primera frase concluimos:

en las cabezas de $P_2 P_3 \ldots P_n$ existe un sombrero blanco (2)

Utilizando lo anterior es fácil probar que el sombrero de P_n es blanco **EJERCICIO**

En la isla de los Caballeros y los Pícaros

En esta isla habitan caballeros (sinceros) y pícaros (mentirosos). Ocasionalmente puede visitar la isla los espías (a veces mienten y otras no).

26 Me encuentro con A,B y C. Pregunto a A: ¿Eres Caballero o Pícaro? La respuesta de A es ininteligible, y pregunto a B: ¿Qué ha dicho A?

B.— A dice que es Pícaro.

C.— No creas a B, que está mintiendo.

¿Qué dijo A? ¿Qué son A y B?

40 Me encuentro con dos habitantes que discuten:

A.— No eres un Caballero.

B.— Tu sí que no lo eres.

Demuestra que uno de ellos es espía, y uno de ellos sincero.

Diálogos Haskell en la Isla

✓ La ejecución de los diálogos-HASKELL ¡son demostraciones!

✓ Ray resuelve los mismos problemas en al menos media página.

```
data Habitante = Pícaro | Caballero | Espía deriving (Eq, Show)
type Par = (Habitante, Habitante)
type Trío = (Habitante, Habitante, Habitante)
```

— la siguiente función proporciona una interpretación de cada frase

```
valorSi :: Frase a → Habitante → Frase a
```

— notación infija y patrones

```
f `valorSi` Caballero = f
f `valorSi` Pícaro    = not . f      {not True = False; not False = T
f `valorSi` Espía     = const True {const k x = k }
```

— no tendremos en cuenta lo que dicen los espías

— Problema 26
— ¿Eres Caballero o Pícaro? La respuesta de A es ininteligible,
— y pregunto a B: ¿Qué ha dicho A?
— B.— A dice que es Pícaro.
— C.— No creas a B, que está mintiendo.

```
sol26 :: Habitante → [Trío]
sol26 res = [ t | t@(a, b, c) ← cubo [Pícaro, Caballero],
                valorSi f a t,    — llamada prefija
                valorSi g b t,
                valorSi h c t ]
                  — interpretamos cada frase según la personalidad
  where
    f, g, h     :: Frase Trío
    f (a, b, c) = a == res                          — A.— soy res.
    g (a, b, c) = f (a, b, c) == (a == Pícaro)      — B.— A dice que es
    h           = not . g                           — C.— B está mintie
```

IslaCaballeros > *sol26 Pícaro*

[]

— A no pudo decir que era Pícaro

IslaCaballeros > *sol26 Caballero*

[(*Pícaro*, *Pícaro*, *Caballero*), (*Caballero*, *Pícaro*, *Caballero*)]

— A dijo que era Caballero, B es un Pícaro y C un Caballero.

— Es imposible saber lo que es A pero sí lo que dijo.

— Problema 27
— Pregunto a A: ¿Cuántos caballeros hay entre vosotros?
— La respuesta es ininteligible, y pregunto a B: ¿Qué ha dicho?
— B.— A dice que hay exactamente un caballero entre nosotros.
— C.— No hagas caso a B que está mintiendo.

```
sol27 res = [ t | t@(a, b, c) ← cubo [Pícaro, Caballero],
                  (f `valorSi` a) t,
                  (g `valorSi` b) t,
                  (h `valorSi` c) t ]
 where
  f, g, h     :: Frase Trío
  f (a, b, c) = res == length [ x | x ← [a, b, c], x == Caballero]
  g (a, b, c) = f (a, b, c) == (1 == length [ x |x ← [a, b, c], x == Ca
                  — uno de nosotros es un caballero
  h           = not . g
```

IslaCaballeros > *sol*27 1
[]
— La respuesta no pudo ser 1.
IslaCaballeros > *sol*27 0
[(*Pícaro*, *Pícaro*, *Caballero*)]
IslaCaballeros > *sol*27 2
[(*Pícaro*, *Pícaro*, *Caballero*), (*Caballero*, *Pícaro*, *Caballero*)]
IslaCaballeros > *sol*27 3
[(*Pícaro*, *Pícaro*, *Caballero*)]
IslaCaballeros > *sol*27 4
[(*Pícaro*, *Pícaro*, *Caballero*)]
— Es imposible deducir la personalidad de A.
— Sabemos que (B,C)== (Pícaro,Caballero).

— Problema 36

— ¿Es alguno de vosotros un caballero?.

— A me respondió, y con su respuesta pude saber la solución.

$$sol36\ res\ =\ [\ p\ |\ p@(a,b)\ \leftarrow\ [Pícaro, Caballero]\ `x`\ [Pícaro, Caballero],\ (f\ `valorSi`\ a)\ p\ ]$$

where

$$f\ ::\ Frase\ Par$$

$$f\ (a,b)\ =\ if\ res == "SI"\ then\ a == Caballero\ ||\ b == Caballero\ else\ a == Pícaro$$

```
IslaCaballeros > sol36 "SI"
[(Pícaro, Pícaro), (Caballero, Pícaro), ...
```

— no queda determinada la personalidad de A

```
IslaCaballeros > sol36 "NO"
[(Pícaro, Caballero)]
```

— A debe ser un Pícaro, y B un Caballero

— Problema 40
— Me he informado y hay espías en la isla. Oigo la conversación:
— A.— B es un Caballero.
— B.— A no es un Caballero.
— Demuestra que al menos una de ellas está diciendo la verdad.

EJERCICIO

Alicia en el bosque del olvido

Alicia al entrar en el bosque olvida, entre otras cosas, el día de la semana. En el bosque hay animales con dos tipos de comportamiento: El León miente los Lunes, Martes y Miércoles, mientras que el Unicornio miente los Jueves, Viernes y Sabados.

48 Alicia se encuentra con el León:

León.— Ayer mentí.

León.— Mentiré dentro de tres días.

¿Qué día de la semana es?

48b *Resolveremos un puzzle más general:*

A.— Ayer mentí.

B.— Mentiré dentro de tres días.

¿Qué día de la semana es ¿Qué son A y B suponiendo que cada personaje conoce la personalidad de todos los habitantes del bosque?

48b *Una interesante modificación:*

A.— Ayer mentí.

B.— Mentiré dentro de tres días.

Alicia es incapaz de deducir el día de la semana ¿Qué comportamientos tenían A y B?

Ahora intervienen Leo y Uni (Tweedledum Y Tweedledee en el original), que son hermanos gemelos (indistinguibles) y se comportan como un León y un Unicornio respectivamente.

51 A.— Yo soy Leo.

B.— Yo soy Uni.

¿Puede Alicia adivinar quienes eran?

57 A.— Si yo soy Leo entonces él es Uni.

B.— Si él es Uni entonces yo soy Leo.

¿Quienes eran?

59 El Rey encuentra el sonajero que rompieron los gemelos y Alicia debe encontrar a su propietario. Tras adentrarse en el bosque se encuentra con los gemelos:

Alicia (dirigiéndose a A).— ¿De quién es el sonajero?

A.— El sonajero es de Uni.

Alicia, tras un momento pensativa, dirg. B.— ¿Quién eres tú?

B.— Yo soy Uni.

Alicia no sabía el día de la semana, pero recuerda que no es Domingo. ¿A quíen debe entregar Alicia el sonajero?

62 Alicia (dirigiéndose a A).— ¿Es tuyo este sonajero?

A.— Sí.

Alicia, tras un momento pensativa, dirigiéndose a B.— ¿Es tuyo este sonajero?

B, susurrándole al oido de Alicia. - ...

No se escuchó la respuesta, aunque fue, o bien Sí, o bien No.

Después de escuchar la respuesta, Alicia sabía quien era el propietario del sonajero. ¿A quíen dió el sonajero?

65 El rey le contó a Alicia que los gemelos podían tener un hermano (Duendi) idéntico a ellos, aunque es un duende (siempre miente). Alicia escucha durante un día laborable:

A.— Duendi existe.

B.— Yo existo.

¿Existe realmente Duendi?

El bosque en Haskell

$$\textbf{data}\ Día\ =\ L\,|M\,|X\ |\ J\ |\ V\ |\ S\ |\ D\ \textbf{deriving}\ (Eq, Show, Enum)$$
$$ayer, mañana\ ::\ Día\ \rightarrow\ Día$$
$$ayer\ L\ =\ D$$
$$ayer\ x\ =\ pred\ x$$
$$mañana\ D\ =\ L$$
$$mañana\ x\ =\ succ\ x$$
$$pasadoMañana\ =\ mañana\ .\ mañana$$
$$dentroDeTresDías\ =\ mañana\ .\ mañana\ .\ mañana$$

$$\textbf{data}\ Habitante\ = León\ |\ Unicornio\ |\ Leo\ |\ Uni\ |\ Duendi$$
$$\textbf{deriving}\ (Eq, Show)$$

$$
\begin{aligned}
&valorSi :: Frase\ a \rightarrow (Habitante, Día) \rightarrow Frase\ a \\
&f\ `valorSi`\ (León, d) &&= if\ d\ `elem`\ [L\,..\,X]\ then\ not\ .\ f\ else\ f \\
&f\ `valorSi`\ (Unicornio, d) &&= if\ d\ `elem`\ [J\,..\,S]\ then\ not\ .\ f\ else\ f \\
&f\ `valorSi`\ (Leo, d) &&= f\ `valorSi`\ (León, d) \\
&f\ `valorSi`\ (Uni, d) &&= f\ `valorSi`\ (Unicornio, d) \\
&f\ `valorSi`\ (Duendi, d) &&= not\ .\ f \quad \text{— Duendi siempre miente}
\end{aligned}
$$

$$
\begin{aligned}
dosOradores = [(a, b, d) \mid\ & a \leftarrow [León, Unicornio], \\
& b \leftarrow [León, Unicornio], \\
& d \leftarrow [L\,..\,D]\ ]
\end{aligned}
$$

$$\begin{aligned} sol48 \;=\; [\; cf \mid cf@(a,b,d) \;\leftarrow\; dosOradores,\\ (f\; `valorSi`\; (a,d))\; cf,\\ (g\; `valorSi`\; (b,d))\; cf\;] \end{aligned}$$

where

— A.- Ayer mentí.

$$\begin{aligned} f\;(Le\acute{o}n,\; b, d) &= (ayer\; d)\; `elem`\; [L,M,X]\\ f\;(Unicornio, b, d) &= (ayer\; d)\; `elem`\; [J,V,S] \end{aligned}$$

— B.- Mentiré dentro de tres días.

$$\begin{aligned} g\;(a, Le\acute{o}n\;, d) &= (dentroDeTresD\acute{i}as\; d)\; `elem`\; [L,M,X]\\ g\;(a, Unicornio, d) &= (dentroDeTresD\acute{i}as\; d)\; `elem`\; [J,V,S] \end{aligned}$$

ElBosqueDelOlvido > *sol*48

[(*León*, *León*, *L*), (*León*, *Unicornio*, *L*), (*León*, *Unicornio*, *J*),
(*Unicornio*, *León*, *D*), (*Unicornio*, *Unicornio*, *J*)]

— Si dialogan dos leones el día es Lunes.

— Si dialogan dos Unicornios, el día es Jueves.

— Problema 51

$$leoOuniYdia \;=\; [(a,b,d) \mid (a,b) \leftarrow [Leo, Uni]\; `x`\; [Leo, Uni],\; a \neq b,\; d \leftarrow [L\,..\,D]\,]$$

$$sol51 \;=\; [\, e \mid e@(a,b,d) \leftarrow leoOuniYdia,\; (f\; `valorSi`\; (a,d))\; e,\; (g\; `valorSi`\; (b,d))\; e\,]$$

where

$$f\;(a,b,d) \;=\; a == Leo \quad \text{— A.- Yo soy Leo.}$$
$$g\;(a,b,d) \;=\; b == Uni \quad \text{— B.- Yo soy Uni.}$$

ElBosqueDelOlvido $>$ $sol51$

$[(Leo, Uni, D)]$

— Luego A es Leo y B es Uni, y el día es Domingo.

— Problema 57

$$sol57 = [\, cf \mid cf@(a,b,d) \leftarrow leoOuniYdia,\ (f\ `valorSi`\ (a,d))\ cf,\ (g\ `valorSi`\ (b,d))\ cf$$

where

— A.- Si yo soy el Leo entonces él es Uni.

$$f\ (a,b,d) = a == Leo\ ==>\ b == Uni$$

— B.- Si él es Uni entonces yo soy Leo.

$$g\ (a,b,d) = a == Uni\ ==>\ b == Leo$$

$$ElBosqueDelOlvido > sol57$$

$$[(Leo, Uni, D), (Uni, Leo, D)]$$

— Ambos enunciados son verdaderos, de forma que ambos son
— sinceros y eso ocurre en Domingo.

— Problema 59: ¿De quién es el sonajero?

— Incluimos un nuevo dato en el estado:

type *Propietario* = *Habitante*

EJERCICIO

— Problema 62: ¿Es tuyo este sonajero?

— A.- Sí.

— Dirigiéndose a B.- ¿Es tuyo este sonajero?

— No se escuchó la respuesta, pero Alicia dedujo su propietario.

EJERCICIO

— Problema 65: ¿Existe Duendi?

El rey le contó a Alicia que los gemelos podían tener un hermano (Duendi) idéntico a ellos, aunque es un duende (siempre miente). Alicia escucha durante un día laborable:

A.— Duendi existe.

B.— Yo existo.

¿Existe realmente Duendi? **EJERCICIO**

¿Vive el conde Drácula?

En Transilvania hay vampiros y humanos, y cada uno de ellos puede estar cuerdo o loco.

Los humanos son sinceros y los vampiros embusteros. Por lo contrario, los cuerdos creen que son ciertas las proposiciones ciertas, y falsas las falsas. Sin embargo, los locos creen que las proposiciones falsas son ciertas y las ciertas, falsas. (un humano loco se comporta *aparentemente* como un embustero, pero al contrario que éste, miente sin maldad).

167 Un transilvano dice: O soy humano o estoy cuerdo. ¿Es un vampiro?

170 Le preguntamos a T: ¿Eres un vampiro Loco? ¿Qué respondió si yo adiviné lo que era?

172 Otro dice: Estoy Loco. ¿Es un vampiro?

176 RS entabla una conversación con dos transilvanos:

RS, dirigiéndose a A.— ¿Es B humano?

A.— Eso creo.

RS, dirigiéndose a B.— ¿A es humano?

¿Qué respondió B?

178 RS, dirigiéndose a A: ¿Crees que eres formal? ¿Qué puedo determinar dependiendo de la respuesta?

¿Es posible deducir si Drácula vive a partir de frases como:

179 Si soy humano entonces Drácula vive aún.

180 Si estoy cuerdo entonces Drácula vive.

182 Si soy formal entonces Drácula vive.

190 ¿Qué preguntaremos a un transilvano para que, independientemente de su comportamineto y su respuesta, podamos saber si Drácula vive?

Transilvanos, Drácula y Haskell

```
data HV = Humano | Vampiro deriving (Eq, Show)
data Comportamiento = Cuerdo | Loco deriving (Eq, Show)

type Transilvano = (HV, Comportamiento)

trans :: [Transilvano]
trans = [Humano, Vampiro] `x` [Cuerdo, Loco]

valorSi, interpreta :: Frase a → Transilvano → Frase a

f `valorSi` (Humano, Cuerdo) = f
f `valorSi` (Humano, Loco)   = not . f
f `valorSi` (Vampiro, Cuerdo) = not . f
f `valorSi` (Vampiro, Loco)  = f
interpreta = valorSi
```

```
cuerdo, loco, vampiro, humano :: Frase Transilvano


cuerdo   (_, c) = c == Cuerdo
loco     (_, c) = c == Loco
vampiro  (t, _) = t == Vampiro
humano   (t, _) = t == Humano

sol167 = [ h | h@(t, c) <- trans,
                  — Soy humano o estoy cuerdo
             (humano `o` cuerdo `valorSi` h) h]


o :: Frase a -> Frase a -> Frase a
f `o` g = λ u -> f u || g u
```

```
EnTransilvania > sol167
[(Humano, Cuerdo)]
```

— Problema 170 : ¿Eres un vampiro loco?
— ¿Qué respondió si yo adiviné su tipo?

```
infixl 1 -:>
(-:>) :: Bool -> (a -> Bool) -> a -> Bool
  — selección con guarda

False -:> f = f
True -:> f = not . f

sol170 :: Bool -> [Transilvano]
sol170 res = [ h | h@(t, c) <- trans,
                   (res -:> (vampiro `y` loco) `valorSi` h) h]
```

$EnTransilvania > sol170\ True$
$[(Humano, Loco), (Vampiro, Cuerdo), (Vampiro, Loco)]$
— con esta respuesta no podemos deducir qué era
$EnTransilvania > sol170\ False$
$[(Humano, Cuerdo)]$
— Por tanto, contestó: NO, y era un humano cuerdo

— Problema 172 : T.- Estoy Loco. ¿Qué puede ser?

$$sol172 = [\, h \mid h@(t, c) \leftarrow trans, \\ interpreta\ loco\ h\ h\,]$$

$EnTransilvania > sol172$
$[(Vampiro, Cuerdo), (Vampiro, Loco)]$
— Luego es un Vampiro

— Problema 176
— RS, dirigiéndose a A.- ¿Es B humano?
— A.- Eso creo.
— RS, dirigiéndose a B.- ¿A es humano?
— ¿Qué respondió?

Aceptaremos un *principio fundamental*: "creo" no depende de su cordura. Necesitamos una función para su interpretación:

$$
\begin{aligned}
&creo :: Frase\ a \rightarrow Transilvano \rightarrow Frase\ a \\
&creo\ f\ (Humano, c) = f \\
&creo\ f\ (Vampiro, c) = not\ .\ f
\end{aligned}
$$

$$
\begin{aligned}
sol176\ res\ =\ [\ pt \mid pt@(a,b)\ \leftarrow\ & trans\ `x`\ trans,\\
& (f\ `creo`\ a)\ pt,\\
& (res\ -:>\ g\ `valorSi`\ b)\ pt\ ]
\end{aligned}
$$

where

$$
\begin{aligned}
f\ (t,t') &= humano\ t'\\
g\ (t,t') &= humano\ t
\end{aligned}
$$

$EnTransilvania >\ sol176\ False$

$[\,]$

— tal respuesta es una contradicción

$EnTransilvania >\ sol176\ True$

$[((Humano, Cuerdo), (Humano, Cuerdo)), ((Humano, Cuerdo), ...$

— Luego la respuesta fue SI

Ilustremos ahora la diferencia de interpretación de las frases 'creo X' y 'X'.

— Problema 177

— RS, dirigiéndose a A.- ¿Eres formal?

— ¿Qué puedo determinar dependiendo de la respuesta?

```
formal :: Frase Transilvano
formal = (== (Humano, Cuerdo)) `o` (== (Vampiro, Loco))

sol177 res = [ a | a <- trans,
                   (res -:> formal `valorSi` a) a ]
```

```
EnTransilvania> sol177 False
[]
```

— Luego, todos los transilvanos contestan SI, y además:

```
EnTransilvania> sol177 True
[(Humano, Cuerdo), (Humano, Loco), (Vampiro, Cuerdo), (Vampiro, Loco
```

— ¡las 4 posibilidades!: No podemos inferir NADA.

— Problema 182

— T.- Si soy formal entonces Drácula vive.

EJERCICIO

— Problema 190:

— ¿Qué preguntar a un transilvano para saber si Drácula vive?

EJERCICIO

Los Cofres de Porcia

Los siguientes problemas están inspirados en Porcia (personaje de *El mercader de Venecia*, de Shakespeare), una princesa que decide elegir esposo, no atendiendo a su belleza o bondad, sino a su agudeza mental o inteligencia.

Porcia tiene varios cofres (de oro, de plata, ...algunos fabricados por Bellini, ...). Cada cofre tiene una inscripción (p.e., diciendo su contenido). Porcia esconde su retrato en un cofre. El pretendiente debe descubrir el cofre que lo contiene de acuerdo con el *diálogo* dado en las inscripciones.

67a

68a – a lo sumo una inscripción de cada cofre es falsa

oro

1 *el retrato no está aquí*
2 *el artista que hizo el retrato es veneciano*

plata

1 *el retrato no está en el de oro*
2 *el artista que hizo el retrato es florentino*

plomo

1 *el retrato no está aquí*
2 *el retrato está en el cofre de plata*

69b

oro

el retrato no está aquí

plata

exactamente uno de estos cofres lo hizo Bellini

70a

oro

el retrato no está aquí

plata

exactamente una de estas inscripciones es cierta

Las inscripciones en Haskell

```
data Color = Oro | Plata | Plomo deriving (Eq, Show)
data Pintor = Veneciano | Florentino deriving (Eq, Show)

valor :: Frase a → Bool → Frase a

f `valor` True  = f
f `valor` _     = not . f

sol67a = [ cont |
           cont ← [Oro, Plata, Plomo],
             — a lo sumo uno de las inscripciones es cierta
           (i_oro, i_plata, i_plomo) ← [(False, False, False), (False, False, Tru
                                        (False, True, False), (True, False, False
           (oro `valor` i_oro) cont,
           (plata `valor` i_plata) cont,
           (plomo `valor` i_plomo) cont] where ...
```

```
oro, plata, plomo :: Frase Color
oro c    = c == Oro    — el retrato está en este cofre
plata c  = c ≠ Plata   — el retrato no está aquí
plomo c  = c ≠ Oro     — el retrato no está en el cofre de oro
```

```
CofresDePorcia > sol67a
[Plata]
```

— El retrato está en el cofre de Plata.

Ray Smullyan expone un razonamiento simple:

Las inscripciones del de oro y el de plomo son contrarias, luego uno de ellas es cierta. Como a lo sumo una de las inscripciones es cierta, la del cofre de plata es falsa, y por tanto el retrato está en el de plata.

```
sol68a = [ (contenedor, pintor) |
    e@(contenedor, pintor) ← [Oro, Plata, Plomo] `x` [Veneciano, Flor
    (i_oro1, i_oro2) ← [(True, True), (True, False), (False, True)],
      — a lo sumo una inscripción es falsa
    (oro1 `valor` i_oro1) e, (oro2 `valor` i_oro2) e,
    (i_plata1, i_plata2) ← [(True, True), (True, False), (False, True)],
    (plata1 `valor` i_plata1) e, (plata2 `valor` i_plata2) e,
    (i_plomo1, i_plomo2) ← [(True, True), (True, False), (False, True)
    (plomo1 `valor` i_plomo1) e, (plomo2 `valor` i_plomo2) e ]
  where {
    oro1, oro2, plata1, plata2, plomo1, plomo2 :: Frase(Color, Pintor);
    oro1 (c, a)    = c ≠ Oro;    oro2 (c, a)    = a == Veneciano;
    plata1 (c, a)  = c ≠ Oro;    plata2 (c, a)  = a == Florentino
    plomo1 (c, a)  = c ≠ Plomo;  plomo2 (c, a)  = c == Plata }
```

CofresDePorcia > *sol*68*a*

[(*Plata*, *Veneciano*), (*Plata*, *Florentino*)]

— El retrato está en el cofre de Plata.

— Su pintor no está determinado.

$sol69b\ =$

EJERCICIO

— ¡la información inicial del problema es esencial

type $Inscripción\ =\ Bool$

$sol70a\ =$

EJERCICIO

Números en la frente (J. Conway)

John elige dos números $p, b \in [1..9]$ y un tercero $y \in [2..18]$

escribe p en la frente de *Paco* y b en la de *Blas*

escribe en una pizarra y junto a la suma $p + b$

Manuel desde otra habitación escucha el siguiente diálogo:

Blas .— *No sé el número de mi frente.*

Paco .— *Yo tampoco.*

Blas .— *Pues sigo igual.*

Paco .— *Y yo.*

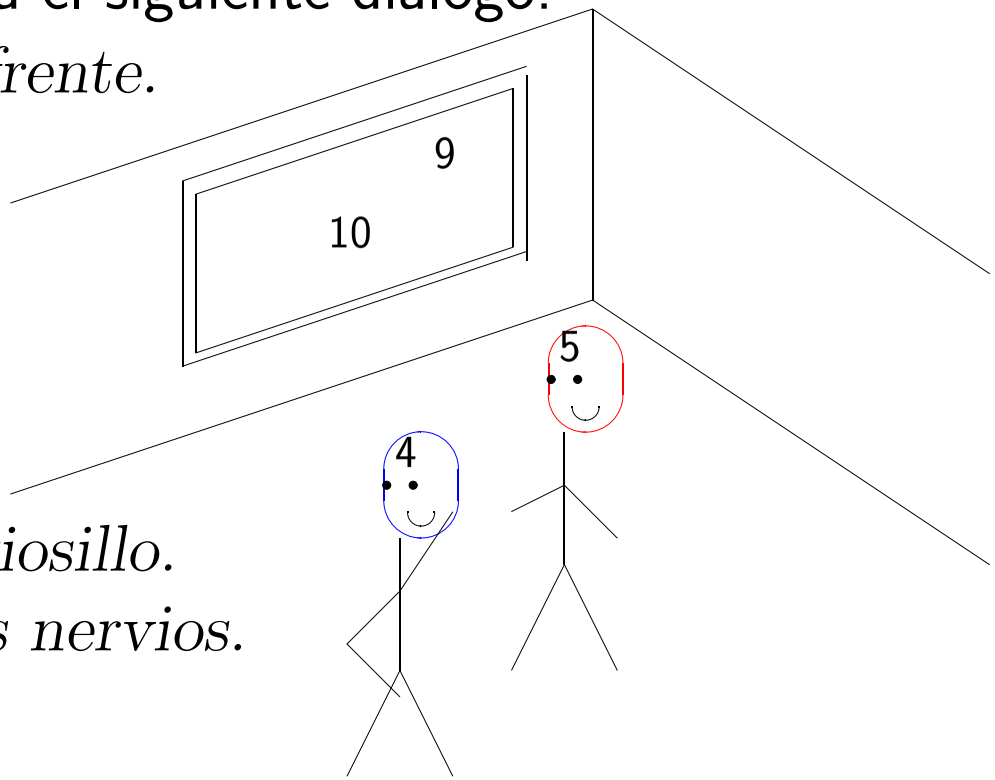
Blas .— *¡Casi lo sé!*

Paco .— *Y yo.*

Blas .— *Me estoy poniendo nerviosillo.*

Paco .— *Yo también estoy de los nervios.*

Blas .— *¡Ajá, ya sé el mío!*



✓ ¿Tiene sentido el diálogo? ✓ ¿sobra la última afirmación?

✓ ¿Puede *Manuel* adivinar los números?

Una plantilla *Haskell*

Este problema es algo más general. Clasificamos la información:

información global	t	9
información pública	la pizarra	10 9
información privada	p, b	4, 5

✓ información privada: la que no tiene el otro;

✓ simetría: *John* reparte b y p, . . . , Blas .— *No sé tu número.*

En general un algoritmo inteligente tiene el tipo :

$$veoYcreo :: InfPrivada \rightarrow InfPublica \rightarrow [InfPrivada]$$

— No se tu dato:

$$f\ miDato\ infPública = varios\ (veoYcreo\ miDato\ infPública)$$

$t = 9$ — información global

data *Pizarra* = *P Int Int* — los números de la pizarra
type *Frente* = *Int* — el número en la frente

— un algoritmo *inteligente*

$$veoYcreo :: Frente \to Pizarra \to [Frente]$$

$$veoYcreo\ f\ (P\ x\ y) = [\, f' \mid f' \leftarrow [x - f, y - f],\ 0 < f',\ f' \leq t]$$

```
EnLaFrente > veoYcreo 3 (P 4 1)
[1]
EnLaFrente > veoYcreo 3 (P 4 6)
[1,3]
```

$$blas,\ paco\ ::\ NdeOrden\ \rightarrow\ Frente\ \rightarrow\ Pizarra\ \rightarrow\ Bool$$

$$\begin{array}{lll} blas\ 1 & p\ piz & = varios\ (veoYcreo\ p\ piz) \\ blas\ (n+1) & p\ piz & = varios\ [\ b \mid b \leftarrow\ veoYcreo\ p\ piz,\ paco\ n\ b\ piz] \\ paco\ n & b\ piz & = varios\ [\ p \mid p \leftarrow\ veoYcreo\ b\ piz,\ blas\ n\ p\ piz] \end{array}$$

$$\begin{array}{l} espacio\ =\ [(p,\ b,\ P\ x\ y) \mid p \leftarrow\ [1..t],\ b \leftarrow\ [1..t], \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathbf{let}\ x\ =\ p+b,\ y \leftarrow\ [2..2*t],\ y\ \neq\ x] \end{array}$$

— búsqueda hacia atrás

$$hastaBlas\ ::\ NdeOrden\ \rightarrow\ [\ (Frente, Frente, Pizarra)\ ]$$

$$\begin{array}{l} hastaBlas\ n\ =\ (\lambda\ (p, b, piz)\ \rightarrow\ blas\ n\ p\ piz)\ `filter`\ espacio \\ hastaPaco\ n\ =\ filter\ (\lambda\ (p, b, piz)\ \rightarrow\ paco\ n\ b\ piz)\ espacio \end{array}$$

Tamaño del Problema

– Tamaño del *espacio de búsqueda* :

$$\begin{array}{cccc} t & \cdot\ t & \cdot\ 2t-2 & \simeq 2t^3 \\ \uparrow & \uparrow & \uparrow & \\ p & b & (x \neq) y \in [2..2t] & \end{array}$$

✓ ¿cuántos elementos filtra la primera afirmación?

$$lenght\ (hastaBlas\ 1)$$

– Tal valor es la mitad del inicial: $t^2(t-1)$

En efecto, si $p, b \in [1..t]$, $x = p + b$,

$$blas\ 1\ p\ (P\ x\ y) \quad \equiv \quad \begin{array}{l} 1 \leq y - p \leq t \\ y \in [p+1..p+t] \end{array} \quad , x \neq y$$

y puede tomar solamente $t-1$ valores.

✓ ¿cuántos elementos filtra la segunda afirmación?

$$lenght\ (hastaPaco\ 1)$$

Una conjetura sobre la solución del problema

La evaluación de $lenght(hastaBlas\ n)$ y ... proporciona:

t	les	hB_1	hP_1	hB_2	hP_2	hB_3	hP_3	hB_4	hP_4	hB_5	hP_5
3	36	18	4								
4	96	48	12	4							
5	200	100	30	12	4						
6	360	180	56	24	12	4					
7	588	294	98	46	24	12	4				
8	896	448	152	76	40	24	12	4			
9	1296	648	228	114	66	40	24	12	4		
10	1800	900	320	168	100	60	40	24	12	4	

...

Conjetura:

en cierto instante el espacio se reduce a 4 combinaciones y después es vacío

– A través del programa podemos obtener tales combinaciones,

t $hastaPaco\ \frac{t-1}{2}$

3	$(1, \boxed{2}, P\ 3\ 4)$	$(2, \boxed{2}, P\ 4\ 3)$	$(2, \boxed{2}, P\ 4\ 5)$	$(3, \boxed{2}, P\ 5\ 4)$
5	$(2, \boxed{3}, P\ 5\ 6)$	$(3, \boxed{3}, P\ 6\ 5)$	$(3, \boxed{3}, P\ 6\ 7)$	$(4, \boxed{3}, P\ 7\ 6)$
7	$(3, \boxed{4}, P\ 7\ 8)$	$(4, \boxed{4}, P\ 8\ 7)$	$(4, \boxed{4}, P\ 8\ 9)$	$(5, \boxed{4}, P\ 9\ 8)$
9	$(4, \boxed{5}, P\ 9\ 10)$	$(5, \boxed{5}, P\ 10\ 9)$	$(5, \boxed{5}, P\ 10\ 11)$	$(6, \boxed{5}, P\ 11\ 10)$

Conjetura:

Para t impar $= 2n+1$ después de la n-ésima afirmación de Paco *, el único número posible para* Blas *es $b = n+1$. La única afirmación con sentido por parte de* Blas *si prosigue el diálogo es: ¡Ajá, ya sé mi número!*

t	$hastaBlas\ \frac{t}{2}$			
4	$(2, 2, P\ 4\ 5)$	$(3, 2, P\ 5\ 6)$	$(2, 3, P\ 5\ 4)$	$(3, 3, P\ 6\ 5)$
6	$(3, 3, P\ 6\ 7)$	$(4, 3, P\ 7\ 8)$	$(3, 4, P\ 7\ 6)$	$(4, 4, P\ 8\ 7)$
8	$(4, 4, P\ 8\ 9)$	$(5, 4, P\ 9\ 10)$	$(4, 5, P\ 9\ 8)$	$(5, 5, P\ 10\ 9)$
10	$(5, 5, P\ 10\ 11)$	$(6, 5, P\ 11\ 12)$	$(5, 6, P\ 11\ 10)$	$(6, 6, P\ 12\ 11)$

Conjetura:

Para $t = 2n$, después de la n-ésima afirmación de Blas *, el número de* Blas *no está determinado. Si prosigue el diálogo el problema no tiene solución (con $2n$ frases).*

Demostración de la conjetura

– Utilizaremos la notación habitual de la matemática

$\mathcal{B}_n(p,x,y)$ y $\mathcal{P}_n(b,x,y)$ predicados hasta la frase n-ésima

$\mathcal{I} = [1..t] \subseteq \mathbb{N}$,

Lema 0.-1.1 *Siendo* $p, b \in \mathcal{I}$; $x, y \in [2..2t]$; $x \neq y$:

$$\begin{array}{lll} \mathcal{B}_1(p,x,y) & \equiv & x-p, y-p \in \mathcal{I} \\ \mathcal{B}_{n+1}(p,x,y) & \equiv & \mathcal{B}_1(p,x,y) \wedge \mathcal{P}_n(x-p,x,y) \wedge \mathcal{P}_n(y-p,x,y) \\ \mathcal{P}_n(b,x,y) & \equiv & \mathcal{B}_1(p,x,y) \wedge \mathcal{B}_n(x-b,x,y) \wedge \mathcal{B}_n(y-b,x,y) \end{array}$$

Teorema 0.-1.2 *Si* $p, b \in \mathcal{I}$, $x, y \in [2..2t]$, $x \neq y$, *y si además* $x = p+q \vee y = p+q$, *entonces:*

$$\begin{array}{lllll} (1) & \textit{Si } t = 2n+1, & \mathcal{P}_n(b,x,y) & \Rightarrow & b = n+1 \wedge |x-y| = 1 \\ (2) & \textit{Si } t = 2n, & \mathcal{B}_n(p,x,y) & \Rightarrow & |x-y| = 1, \neg\mathcal{P}_n(b,x,y) \\ & & & & p, b \in \{n, n+1\} \end{array}$$

Para $t = 9$, $n = 4$ (8 frases), *Manuel* sabrá que $b = 5$.

Variantes del problema

Variante 1

John toma los números del intervalo $\mathcal{I} = [1..t]$, y el diálogo es:

Blas .— *No conozco el número de mi frente.*
Paco .— *Yo tampoco.*
Blas .— *Pues, sigo sin saberlo.*
Paco .— *Entonces, yo tampoco.*
Blas .— *Ya lo sé*

¿Puede *Manuel* adivinar el valor de t? ¿y los números?

EJERCICIO

Variante 2

Manuel conoce el valor de t, y el diálogo es:

Blas .— *No conozco el número de mi frente.*
Paco .— *Yo tampoco.*
Blas .— *Pues, sigo sin saberlo.*
Paco .— *Entonces, yo tampoco.*
Manuel .— *¡Tarará que te ví! Ya sé el número de* Blas

¿Cual es el valor de t? **EJERCICIO**
¿Y si *Paco* y *Blas* no conocen t? ¿Es trascendente? **EJERCICIO**

El *Problema P-S* (H Freudenthal)

Hans elige dos números distintos mayores que 1, entrega el producto a $\mathcal{PROD}$ y la suma a $\mathcal{SUM}$, que mantienen el siguiente diálogo:

$\mathcal{SUM}$.— *No se cómo vas a adivinar mi suma.*

$\mathcal{PROD}$.— *Entonces, no sé tu suma*

$\mathcal{SUM}$.— *Pues yo ya se tu producto.*

¿Cuáles eran los números elegidos por Hans?

✓ En el problema original (1969) la segunda frase es:

$\mathcal{PROD}$.— *¡Ajá!, entonces ya sé tu suma*

información global	$x, y > 1,\ x \neq y$
información pública	Φ
información privada	s, p

Un programa *Haskell*

$$
\begin{aligned}
sum\ \ s &= and\ (\ map\ varios\ [\_S\ p \mid p \leftarrow \_P\ s]\ ) \\
sum &= and\ .\ map\ varios\ .\ map\ \_S\ .\ \_P \\
pro\ \ p &= varios\ [s \mid s \leftarrow \_S\ p,\ sum\ s] \\
pro &= varios\ .\ filter\ sum\ .\ \_S \\
sum'\ s &= uno\ [p \mid p \leftarrow \_P\ s,\ pro\ p] \\
sol &= [s \mid s \leftarrow [5..], sum'\ s] \\
\_P\ \ s &= [x * (s - x) \mid x \leftarrow [2..(s-1)\ `div`\ 2]] \\
\_S\ \ p &= \ldots
\end{aligned}
$$

✓ evaluando tenemos $sol = [11,\qquad\qquad (p = 30)$

✓ En el problema original (1969) la segunda frase es:

PROD.— ¡Ajá!, entonces ya sé tu suma

EJERCICIO