



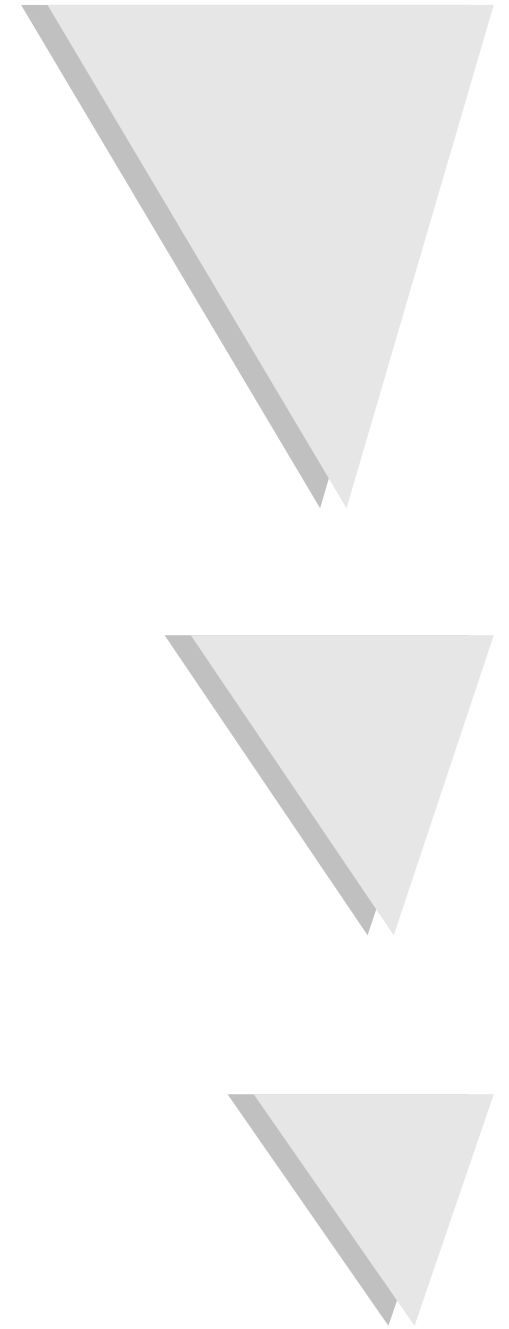
PROGRAMACIÓN FUNCIONAL

Técnicas Formales: Propiedades y Demostraciones



Técnicas Formales

- ◆ Propiedades y demostraciones
 - ◆ Definición de propiedad y demostración
 - ◆ Demostraciones formales
 - ◆ Formas de garantizar propiedades
 - ◆ demostración manual
 - ◆ demostración automática
 - ◆ por construcción
 - ◆ Ejemplos



Propiedad

◆ Sentencia sobre un elemento o conjunto, que puede ser verdadera o falsa.

◆ Ejemplos:

2 es un número primo

(4 = doble 2)

($\exists x \in \mathbb{N}$. x es par)

($\exists k \in \mathbb{N}$. $\sum_{i=0, \dots, k} (2 \cdot i + 1) \neq k^2$)

Propiedades

- ◆ ¿Cómo garantizar una propiedad?
 - ◆ Demostración manual
 - ◆ de manera informal (pej. argumento relatado)
 - ◆ de manera formal (pej. cadena de igualdades justificadas)
 - ◆ en papel o asistido por computadoras
 - ◆ Automáticamente
 - ◆ un programa verifica la propiedad (pej. inferencia de tipos)
 - ◆ Por construcción
 - ◆ los elementos se construyen de tal manera que se cumpla la propiedad (pej. derivación de programas)

Demostración

- ◆ Argumentación que hace evidente la verdad de una propiedad.
- ◆ Ejemplos:
 - ◆ $\text{doble } 2 = 2 + 2 = 4$
 ↑ ↑
 por def. de doble arit.
 - ◆ Dado que 2 es un número natural, y es par, entonces es claro que es verdadera la propiedad $(\exists x \in \mathbb{N}. x \text{ es par})$

Demostraciones

- ◆ ¿Cómo escribir una demostración?
 - ◆ Informalmente
 - ◆ se argumenta describiendo los pasos a seguir para que la propiedad sea evidente.
 - ◆ Formalmente
 - ◆ se utiliza un lenguaje formal (por ejemplo, lógica) para construir una cadena de pasos evidentes
- ◆ Ejemplo: demostrar que
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Informalmente

Primero mostramos que $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$. Si $x \in A \cup (B \cap C)$, entonces, o bien $x \in A$, o bien $x \in B \cap C$. Si $x \in A$, es claro que $x \in A \cup B$ y que $x \in A \cup C$, por lo que $x \in (A \cup B) \cap (A \cup C)$. Por otro lado, si $x \in B \cap C$, entonces $x \in B$ y $x \in C$, por lo que $x \in A \cup B$ y que $x \in A \cup C$, y nuevamente $x \in (A \cup B) \cap (A \cup C)$.

Recíprocamente, si $y \in (A \cup B) \cap (A \cup C)$, entonces $y \in A \cup B$ y $y \in A \cup C$.

Consideramos dos casos: $y \in A$ e $y \notin A$. Si $y \in A$ es claro que $y \in A \cup (B \cap C)$, y esta parte está lista. Si $y \notin A$, entonces, como $y \in A \cup B$, tenemos que $y \in B$, y de la misma manera, $y \in C$. Por lo tanto, $y \in B \cap C$, y eso implica $y \in A \cup (B \cap C)$. Ambas partes muestran que $A \cup (B \cap C) \supseteq (A \cup B) \cap (A \cup C)$.

La propiedad queda entonces demostrada.

Formalmente

$$\begin{aligned} & x \in A \cup (B \cap C) \\ \equiv & \quad \quad \quad (\text{def. de unión}) \\ & x \in A \vee x \in (B \cap C) \\ \equiv & \quad \quad \quad (\text{def. de intersección}) \\ & x \in A \vee (x \in B \wedge x \in C) \\ \equiv & \quad \quad \quad (\text{distributividad de conectivos lógicos}) \\ & (x \in A \vee x \in B) \wedge (x \in A \vee x \in C) \\ \equiv & \quad \quad \quad (\text{def. de unión, dos veces}) \\ & (x \in A \cup B) \wedge (x \in A \cup C) \\ \equiv & \quad \quad \quad (\text{def. de intersección}) \\ & x \in (A \cup B) \cap (A \cup C) \end{aligned}$$

Demostraciones formales

- ◆ Se comienza con una expresión, y se la transforma paso a paso.
- ◆ Cada nueva versión de la expresión se escribe en una línea nueva.
- ◆ Entre dos líneas se coloca un símbolo que relaciona las expresiones, y una justificación de la validez del paso.
- ◆ El nivel de detalle puede variar.
 - ◆ En el ejemplo, la penúltima línea podría hacerse en dos pasos.

Ventajas

- ◆ Las pruebas formales fuerzan a que la estrategia de demostración sea explícita.
- ◆ La solución es fácil de leer y de verificar.
- ◆ Puede refinarse una demostración, agregando detalles.
- ◆ Pueden desarrollarse programas que ayuden a construir, chequear y revisar pruebas de este tipo.

Principio de Extensionalidad

- ◆ La extensionalidad nos permite demostrar que $f = g$ demostrando simplemente que $f\ x = g\ x$ para todo x .
- ◆ Dicho de otro modo,
 - ◆ $f = g$ sí y sólo sí para todo x , $f\ x = g\ x$.
- ◆ Ej: ¿Se cumple $\text{doble} = \text{doble}'$?
 $\text{doble}, \text{doble}' :: \text{Int} \rightarrow \text{Int}$
 $\text{doble}\ x = x + x$
 $\text{doble}'\ x = 2 * x$

Ejemplo (1 de 2)

◆ **Propiedad:** $\text{curry suma}' = \text{suma}$

◆ **Demostración:**

◆ demostramos que para x e y cualesquiera
 $\text{curry suma}' x y = \text{suma } x y$

Dado que dos funciones son iguales si aplicadas a los mismos argumentos dan los mismos resultados (principio de extensionalidad), entonces concluimos la demostración.

Ejemplo (2 de 2)

Sean $x, y :: \text{Int}$, enteros cualesquiera.

$$\begin{aligned} & \text{curry suma}' x y \\ = & \text{suma}' (x,y) && (\text{def. de curry}) \\ = & x+y && (\text{def. de suma}') \\ = & \text{suma } x y && (\text{def. de suma}) \end{aligned}$$

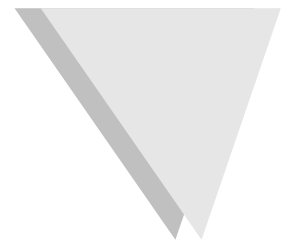
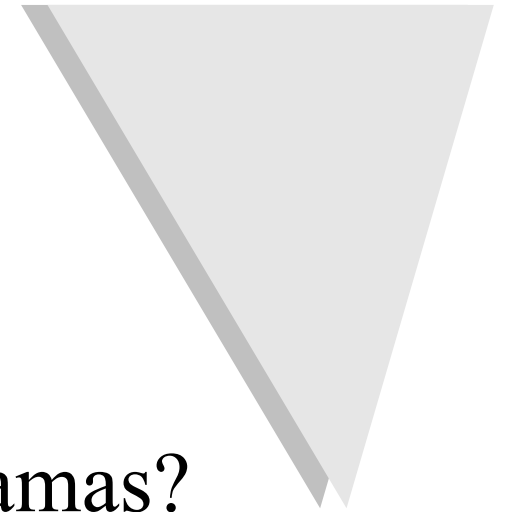
Como la igualdad vale, la propiedad se cumple.

Propiedades

- ◆ ¿Qué propiedades de los programas nos interesan en este curso?
 - ◆ Equivalencia
 - ◆ si f y g son dos programas, queremos saber si $f = g$
 - ◆ Corrección
 - ◆ el programa hace lo esperado (por ejemplo, no da error, o devuelve el resultado correcto)
 - ◆ Terminación
 - ◆ el programa no se queda realizando infinitas reducciones

Propiedades

- ◆ ¿Cómo probar equivalencia de programas?
 - ◆ Usando las ecuaciones del script
 - ◆ Usando propiedades ya probadas
- ◆ ¿Cómo garantizar corrección?
 - ◆ Usando el sistema de tipos
 - ◆ Usando técnicas de derivación de programas
 - ◆ Diseñando casos disjuntos y completos
- ◆ ¿Cómo garantizar terminación?
 - ◆ Evitando ciclos infinitos

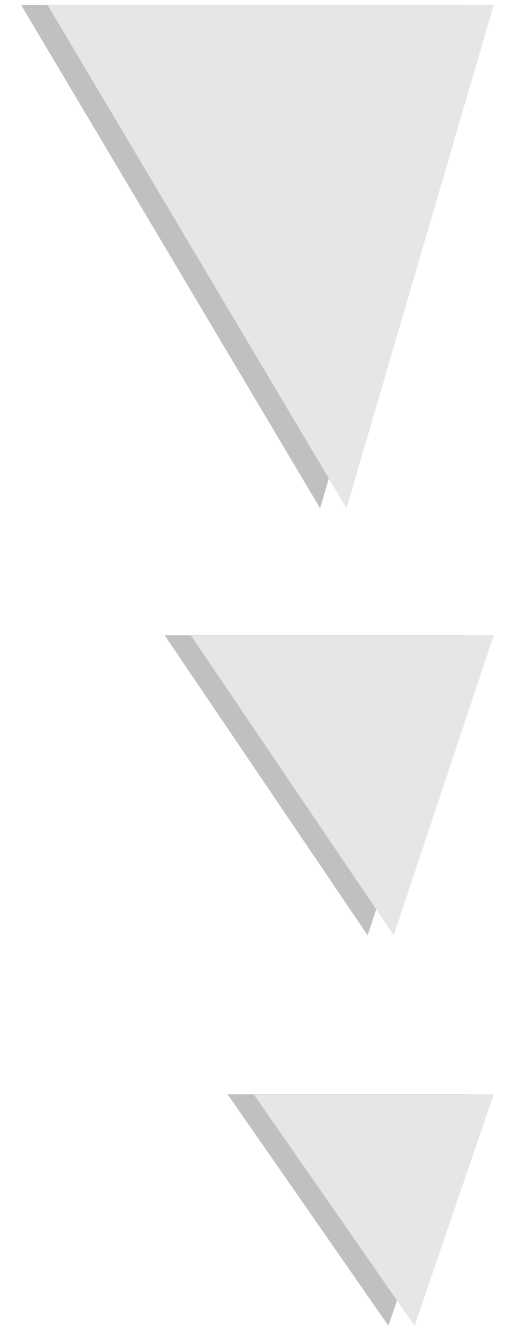


Propiedades

- ◆ Diseñar casos disjuntos y completos

- ◆ Ejemplo: $\text{par, impar} :: \text{Int} \rightarrow \text{Bool}$
 $\text{par } 0 = \text{True}$
 $\text{par } n \mid n > 0 = \text{impar } (n-1)$
 $\text{par } n \mid n < 0 = \text{impar } (n+1)$
 $\text{impar } 0 = \text{False}$
 $\text{impar } n \mid n > 0 = \text{par } (n-1)$
 $\text{impar } n \mid n < 0 = \text{par } (n+1)$

- ◆ Así definidas, son totales.



Terminación

◆ ¿Cuándo aparecen ciclos infinitos?

◆ Ej:

factorial :: Int -> Int

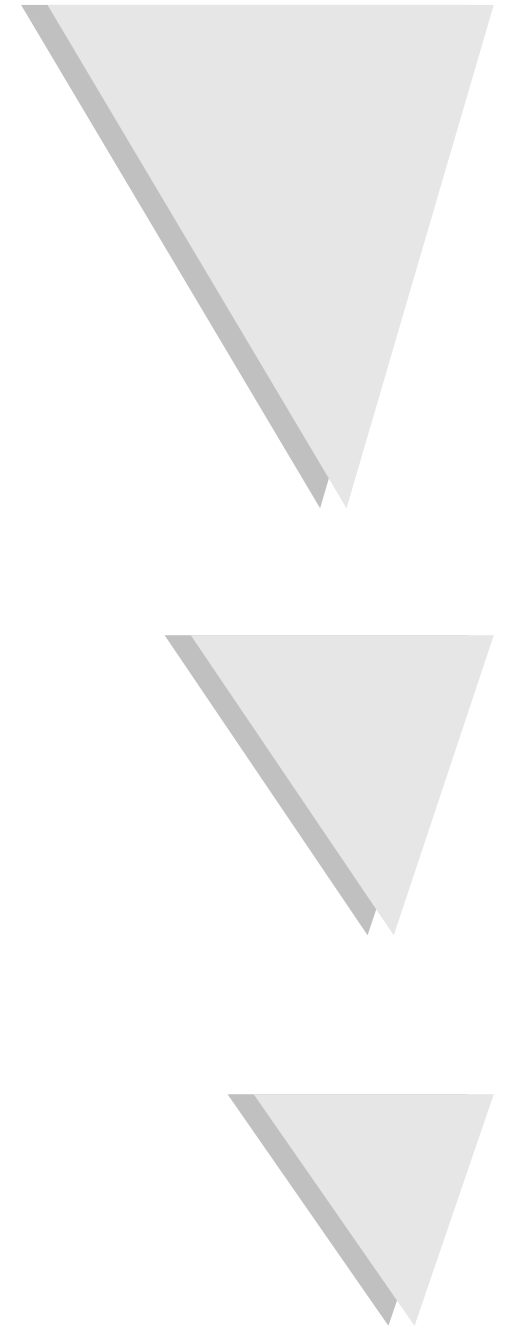
factorial 0 = 1

factorial n | n /= 0 = n * factorial (n-1)

◆ ¿Cuánto vale factorial (-1)?

◆ ¿Son ecuaciones orientadas?

◆ ¿Cómo saber si lo son o no?



Equivalencia

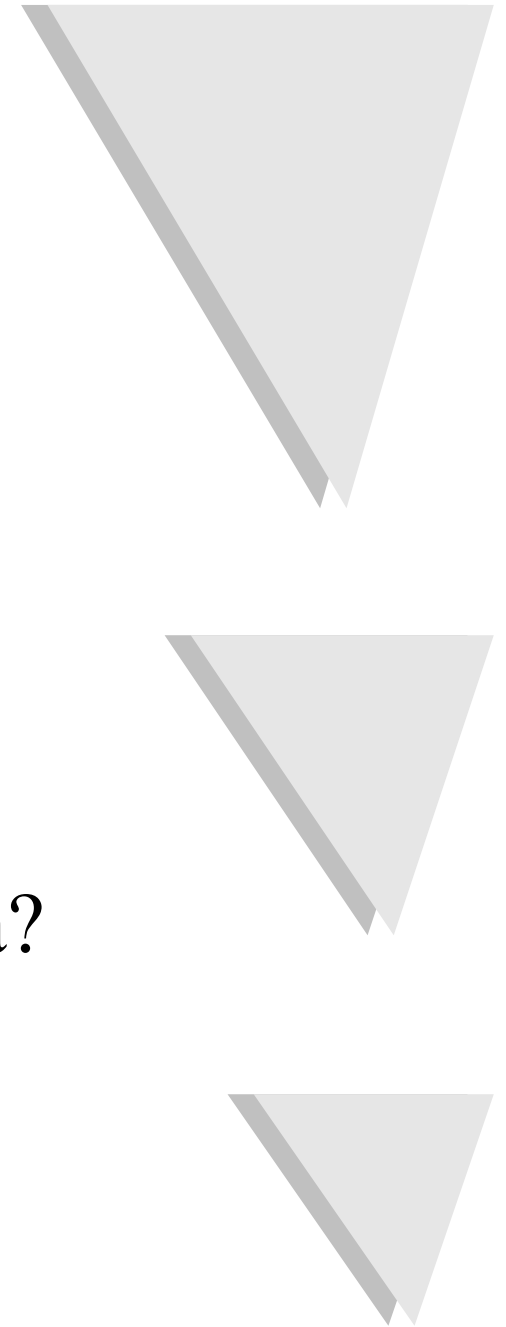
- ◆ Considere esta definición

$\text{fact} :: \text{Int} \rightarrow \text{Int}$

$\text{fact } 0 = 1$

$\text{fact } n \mid n \neq 0 = \text{fact } (n-1) * n$

- ◆ ¿Es equivalente a la definición previa?
 - ◆ O sea, ¿se cumple $\text{factorial} = \text{fact}$?
 - ◆ ¿Podría demostrarlo? ¿Cómo?



Conclusiones

- ◆ El tema cubre:
 - ◆ qué es una propiedad y qué significa demostrar propiedades
 - ◆ cuáles son las propiedades que nos interesan
 - ◆ cómo garantizar algunas propiedades por construcción