Programación Funcional

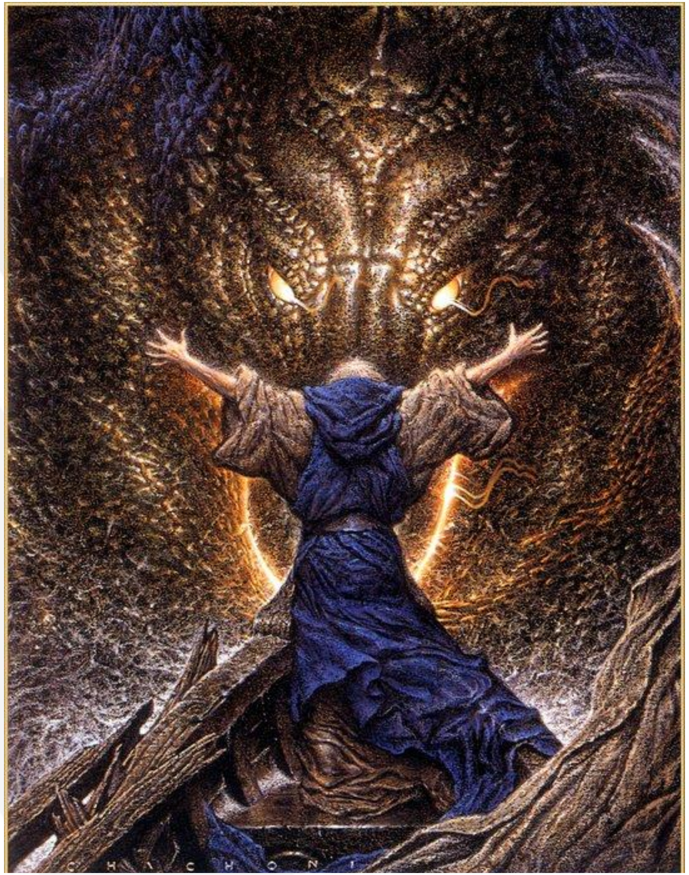
Clases teóricas

por Pablo E. "Fidel" Martínez López

6. Propiedades y demostraciones

"En verdad el mago le había dicho eso una vez, pero Ged no le había hecho mucho caso; aunque ahora sabía que Ogión nunca le diría nada sin alguna buena razón."

Un mago de Terramar
Úrsula K. Le Guin



Motivación

Motivación

- Se presentaron visiones denotacional y operacional
- Se presentó la equivalencia (=) entre expresiones
  - Equivalentes significa que tienen el mismo significado
- ¿Cómo podemos estar seguros de que una afirmación de equivalencia es cierta?
  - Es necesario alguna forma de convencerse

Propiedades y demostraciones

Propiedades y demostraciones

- ¿Qué es una ***propiedad*** en este contexto?

Propiedades y demostraciones

- ¿Qué es una ***propiedad*** en este contexto?
  - *Afirmación* sobre un elemento o conjunto que puede ser verdadera o falsa

Propiedades y demostraciones

- ¿Qué es una ***propiedad*** en este contexto?
  - *Afirmación* sobre un elemento o conjunto que puede ser verdadera o falsa
  - Ejemplos:
    - 2 es un número primo
    - `cuadruple 2 = doble (doble 2)`
    - $(\exists x \in \mathbb{N}. (\forall y \in \mathbb{N}. x > y))$

Propiedades y demostraciones

- ¿Qué es una ***propiedad*** en este contexto?
  - *Afirmación* sobre un elemento o conjunto que puede ser verdadera o falsa
  - Ejemplos:
    - ¿2 es un número primo?
    - ¿`cuadruple 2 = doble (doble 2)`?
    - ¿$(\exists x \in \mathbb{N}.\ (\forall y \in \mathbb{N}.\ x > y))$?
  - Para saber si una propiedad es verdadera o falsa, se la puede plantear como una **pregunta** para responder por sí o no

Propiedades y demostraciones

- ¿Qué es una ***demostración*** de una propiedad?

Propiedades y demostraciones

- ¿Qué es una ***demostración*** de una propiedad?
  - *Argumentación* que hace evidente que es verdadera
    - Puede ser algo sencillo, o muy complejo

Propiedades y demostraciones

- ¿Qué es una ***demostración*** de una propiedad?
  - *Argumentación* que hace evidente que es verdadera
    - Puede ser algo sencillo, o muy complejo
  - Ejemplo:
    - ¿2 es un número primo?

Propiedades y demostraciones

- ¿Qué es una ***demostración*** de una propiedad?
  - *Argumentación* que hace evidente que es verdadera
    - Puede ser algo sencillo, o muy complejo
  - Ejemplo:
    - ¿2 es un número primo?
      - *los únicos divisores de 2 son -2, -1, 1 y 2*
      - *por definición de primo, 2 es primo*
      - *La propiedad es verdadera*

Propiedades y demostraciones

- ¿Qué es una ***demostración*** de una propiedad?
  - *Argumentación* que hace evidente que es verdadera
    - Puede ser algo sencillo, o muy complejo
  - Ejemplo:
    - ¿2 es un número primo?
      - *los únicos divisores de 2 son -2, -1, 1 y 2*
      - *por definición de primo, 2 es primo*
      - *La propiedad es verdadera*
  - Si la propiedad es falsa, NO se puede demostrar

Propiedades y demostraciones

- ¿Qué maneras hay de hacer demostraciones?

Propiedades y demostraciones

- ¿Qué maneras hay de hacer demostraciones?
  - De forma manual
    - de manera informal (e.g. argumento relatado)
    - de manera formal (e.g. reglas estrictas de formación)

Propiedades y demostraciones

- ¿Qué maneras hay de hacer demostraciones?
  - De forma manual
    - de manera informal (e.g. argumento relatado)
    - de manera formal (e.g. reglas estrictas de formación)
  - De forma automática (solo para ciertas propiedades)
    - un programa verifica la propiedad (e.g. sistema de tipos)

Propiedades y demostraciones

- ¿Qué maneras hay de hacer demostraciones?
  - De forma manual
    - de manera informal (e.g. argumento relatado)
    - de manera formal (e.g. reglas estrictas de formación)
  - De forma automática (solo para ciertas propiedades)
    - un programa verifica la propiedad (e.g. sistema de tipos)
  - Por construcción
    - se siguen reglas para encontrar los elementos
      (e.g. resolución de ecuaciones, derivación de programas)

Propiedades y demostraciones

- Nos concentraremos en demostraciones manuales de equivalencias de manera formal

Propiedades y demostraciones

- Nos concentraremos en demostraciones manuales de equivalencias de manera formal
  - Se comienza con una expresión

Propiedades y demostraciones

- Nos concentraremos en demostraciones manuales de equivalencias de manera formal
  - Se comienza con una expresión
  - Se la transforma paso a paso a través de reglas conocidas (definiciones, otras propiedades)

Propiedades y demostraciones

- Nos concentraremos en demostraciones manuales de equivalencias de manera formal
  - Se comienza con una expresión
  - Se la transforma paso a paso a través de reglas conocidas (definiciones, otras propiedades)
  - Para presentar, escribir de a una expresión por línea, intercaladas por la relación entre líneas, justificada
    - Usualmente el símbolo es equivalencia

Propiedades y demostraciones

- Nos concentraremos en demostraciones manuales de equivalencias de manera formal
  - Se comienza con una expresión
  - Se la transforma paso a paso a través de reglas conocidas (definiciones, otras propiedades)
  - Para presentar, escribir de a una expresión por línea, intercaladas por la relación entre líneas, justificada
    - Usualmente el símbolo es equivalencia
  - Al terminar, puede (o no) cerrarse con un símbolo o frase

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem:

```
cuadruple 2
  =
    4*2
  =
   4+4
doble 4
 =
doble
=
doble (doble 2)
```

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem:

```
cuadruple 2
=                  (def. de cuadruple, con x = 2)
4*2
=
4+4

doble 4
=
doble
=

doble (doble 2)
```

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem:

```
cuadruple 2
=                    (def. de cuadruple, con x = 2)
4*2
=                    (aritmética)
4+4
=
doble 4
=
doble
=
doble (doble 2)
```

Propiedades y demostraciones

- **Prop.:** ¿`cuadruple 2` = `doble (doble 2)`?
- **Dem:**

```
    cuadruple 2
=                   (def. de cuadruple, con x = 2)
    4*2
=                   (aritmética)
    4+4

    doble 4
    =

    doble (2+2)
=                   (def. de doble, con x = 2)
    doble (doble 2)
```

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem:

```
    cuadruple 2
=                  (def. de cuadruple, con x = 2)
    4*2
=                  (aritmética)
    4+4

    doble 4
=                  (aritmética)
    doble (2+2)
=                  (def. de doble, con x = 2)
    doble (doble 2)
```

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem: `cuadruple 2`

```
    cuadruple 2
=                  (def. de cuadruple, con x = 2)
    4*2
=                  (aritmética)
    4+4
=                  (def. de doble, con x = 4)
    doble 4
=                  (aritmética)
    doble (2+2)
=                  (def. de doble, con x = 2)
    doble (doble 2)
```

Propiedades y demostraciones

- Prop.: ¿`cuadruple 2` = `doble (doble 2)`?
- Dem: `cuadruple 2`

```
cuadruple 2
=               (def. de cuadruple, con x = 2)
4*2
=               (aritmética)
4+4
=               (def. de doble, con x = 4)
doble 4
=               (aritmética)
doble (2+2)
=               (def. de doble, con x = 2)
doble (doble 2)
```

- Observar las equivalencias justificadas
- Recordar que la equivalencia es simétrica y transitiva

Propiedades y demostraciones

- Prop.: ¿`succ (cuadruple 2) = succ (doble (doble 2))`?
- Dem:

Propiedades y demostraciones

- Prop.: ¿`succ (cuadruple 2) = succ (doble (doble 2))`?
- Dem:

```
succ (cuadruple 2)
```

= (prop. anteriormente demostrada -- LEMA)

```
succ (doble (doble 2))
```

Propiedades y demostraciones

- **Prop.:** ¿`succ (cuadruple 2)` = `succ (doble (doble 2))`?
- **Dem:**

```
    succ (cuadruple 2)
=                      (prop. anteriormente demostrada -- LEMA)
    succ (doble (doble 2))
```

  - Observar
    - no es necesario volver a analizar los pasos anteriores
    - no hace falta saber la definición de `succ`

Propiedades y demostraciones

- **Prop.:** ¿`succ (cuadruple 2)` = `succ (doble (doble 2))`?
- **Dem:**

```
    succ (cuadruple 2)
=                      (prop. anteriormente demostrada -- LEMA)
    succ (doble (doble 2))
```

  - Observar
    - no es necesario volver a analizar los pasos anteriores
    - no hace falta saber la definición de `succ`
  - Se puede usar la demostración ya hecha
    - ¡Como si fuese una subtarea (procedimiento)!

Propiedades y demostraciones

- **Prop.:** ¿`succ (cuadruple 2)` = `succ (doble (doble 2))`?
- **Dem:**

```
  succ (cuadruple 2)
=                     (prop. anteriormente demostrada -- LEMA)
  succ (doble (doble 2))
```

  - Observar
    - no es necesario volver a analizar los pasos anteriores
    - no hace falta saber la definición de `succ`
  - Se puede usar la demostración ya hecha
    - ¡Como si fuese una subtarea (procedimiento)!
    - Un ***lema*** es una propiedad que funciona como subtarea

Propiedades y demostraciones

- Ventajas de las demostraciones formales

Propiedades y demostraciones

- Ventajas de las demostraciones formales
  - La *estrategia* de demostración es explícita
    - Los ***lemas*** funcionan como *subtareas*

Propiedades y demostraciones

- Ventajas de las demostraciones formales
  - La *estrategia* de demostración es explícita
    - Los ***lemas*** funcionan como *subtareas*
  - La solución es fácil de leer y verificar

Propiedades y demostraciones

- Ventajas de las demostraciones formales
  - La *estrategia* de demostración es explícita
    - Los ***lemas*** funcionan como *subtareas*
  - La solución es fácil de leer y verificar
  - Puede refinarse agregando detalles

Propiedades y demostraciones

- Ventajas de las demostraciones formales
  - La *estrategia* de demostración es explícita
    - Los ***lemas*** funcionan como *subtareas*
  - La solución es fácil de leer y verificar
  - Puede refinarse agregando detalles
  - Pueden desarrollarse programas que ayuden a construir, chequear y revisar pruebas de este tipo

Equivalencia de funciones

Equivalencia de funciones

- ¿Cómo demotramos equivalencia de funciones?

Equivalencia de funciones

- ¿Cómo demotramos equivalencia de funciones?
  - Por ejemplo
    - ¿`twice doble = cuadruple`?
    - ¿`twice twice = \f x -> f (f (f (f x)))`?
    - ¿`curry suma' = suma`?

Equivalencia de funciones

- ¿Cómo demotramos equivalencia de funciones?
  - Por ejemplo
    - `¿twice doble = cuadruple?`
    - `¿twice twice = \f x -> f (f (f (f x)))?`
    - `¿curry suma' = suma?`
  - Usualmente las definiciones tienen parámetros

Equivalencia de funciones

- ¿Cómo demotramos equivalencia de funciones?
  - Por ejemplo
    - `¿twice doble = cuadruple?`
    - `¿twice twice = \f x -> f (f (f (f x)))?`
    - `¿curry suma' = suma?`
  - Usualmente las definiciones tienen parámetros
  - ¡Falta algo para demostrar equivalencias de funciones!

Equivalencia de funciones

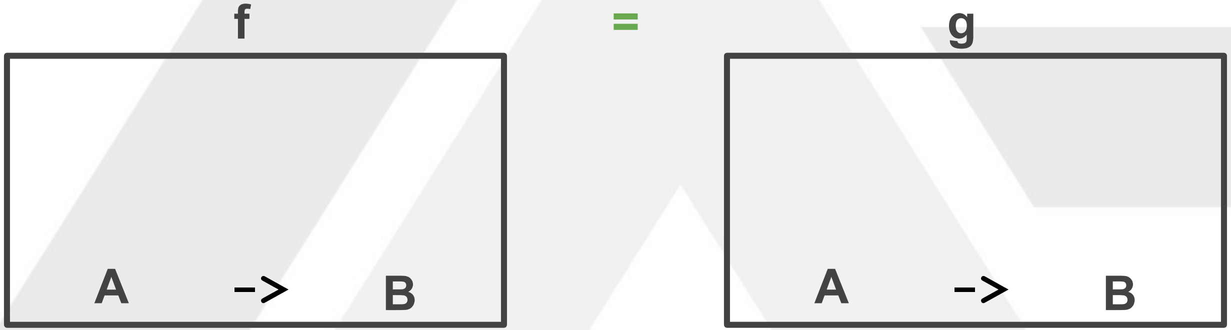
- ¿Cómo saber cuándo dos funciones son la misma?

$$f = g$$

$$A \to B \qquad\qquad A \to B$$

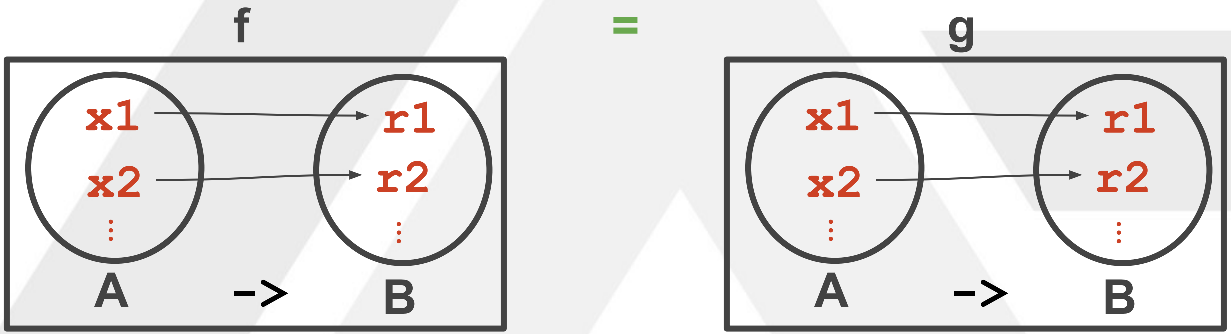
Equivalencia de funciones

- ¿Cómo saber cuándo dos funciones son la misma?
  - Dado que son transformaciones, deberían transformar cada elemento exactamente de la misma manera…



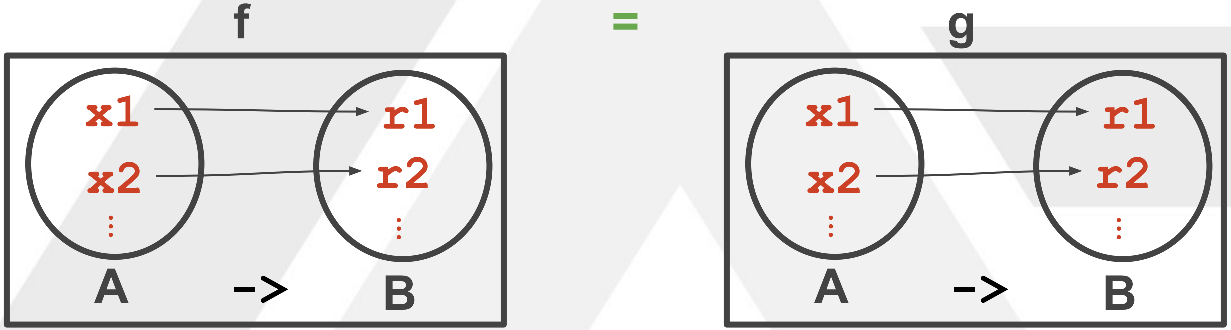
Equivalencia de funciones

- ¿Cómo saber cuándo dos funciones son la misma?
  - Dado que son transformaciones, deberían transformar cada elemento exactamente de la misma manera…



Equivalencia de funciones

- ¿Cómo saber cuándo dos funciones son la misma?
  - Dado que son transformaciones, deberían transformar cada elemento exactamente de la misma manera…



  - ¿Cómo expresamos esto con una propiedad?

Equivalencia de funciones

f = g

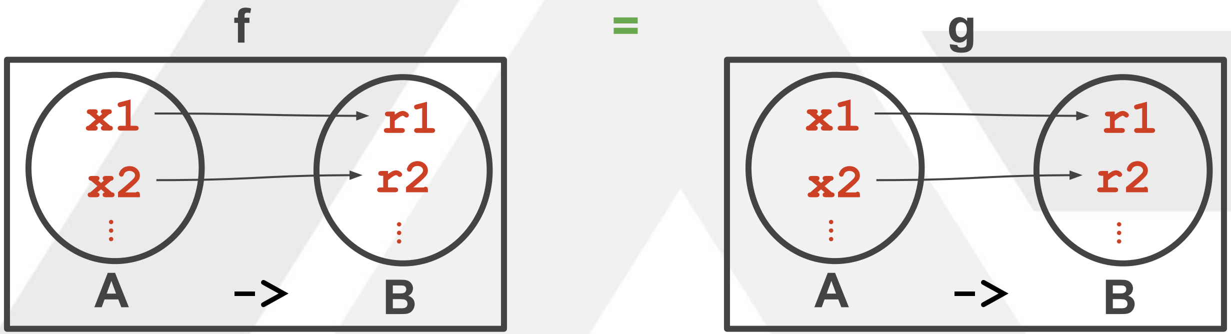
f = g

A -> B

A -> B

Equivalencia de funciones

- **f = g** es equivalente a para todo **x**. **f x = g x**

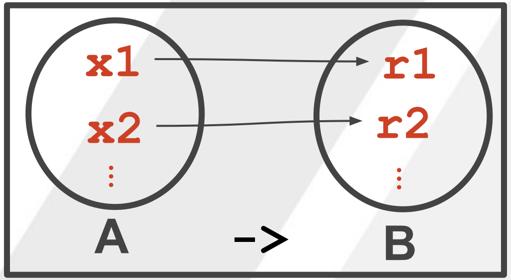


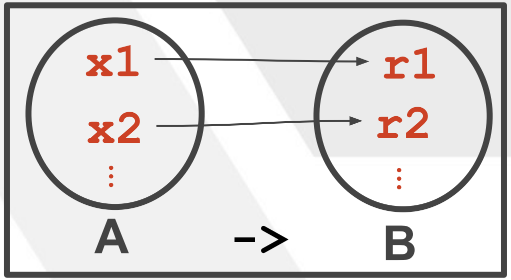
Principio de extensionalidad

- ***Principio de extensionalidad***

  ¿f = g? es equivalente a ¿para todo **x**. **f x** = **g x**?

f = g



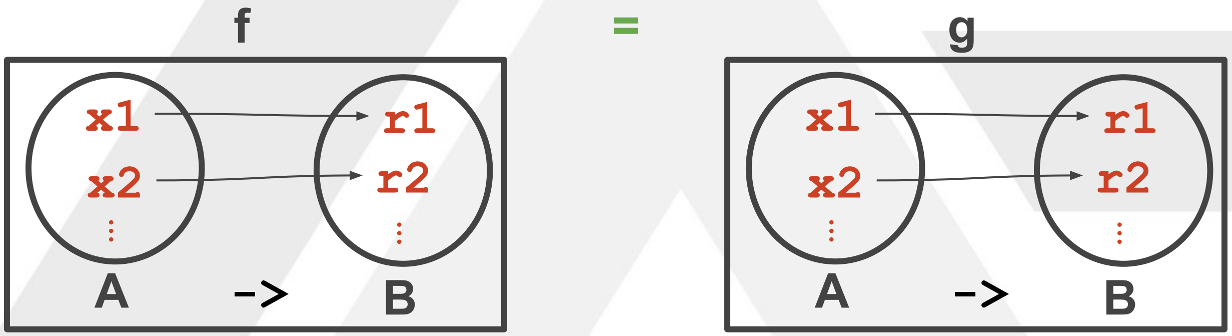


- ¡Ambas preguntas se responden igual!

Principio de extensionalidad

- ***Principio de extensionalidad***

  ¿**f** = **g**? es equivalente a ¿para todo **x**. **f x** = **g x**?



- ¡Ambas preguntas se responden igual!

Principio de extensionalidad

Principio de extensionalidad		
¿**f = g**?	es equivalente a	¿para todo **x**. **f x = g x**?

Principio de extensionalidad

- ¿Cómo utilizar el principio de extensionalidad?
  - Prop.: ¿`twice doble` = `cuadruple`?

    Dem.:

Principio de extensionalidad

- ¿Cómo utilizar el principio de extensionalidad?
  - **Prop.:** ¿`twice doble` = `cuadruple`?

    Dem.: Por el principio de extensionalidad,

    es equivalente demostrar que

    ¿para todo **x**. `twice doble x` = `cuadruple x`?

Principio de extensionalidad

> *Principio de extensionalidad*
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- ¿Cómo utilizar el principio de extensionalidad?
  - **Prop.:** ¿$\overset{f}{\boxed{\texttt{twice doble}}} = \overset{g}{\boxed{\texttt{cuadruple}}}$?

    Dem.: Por el principio de extensionalidad,

    es equivalente demostrar que

    ¿para todo **x**. $\overset{f}{\boxed{\texttt{twice doble}}}\ \mathbf{x} = \overset{g}{\boxed{\texttt{cuadruple}}}\ \mathbf{x}$?

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- ¿Cómo utilizar el principio de extensionalidad?
  - **Prop.:** ¿`twice doble` = `cuadruple`?

    Dem.: Por el principio de extensionalidad,

    es equivalente demostrar que

    ¿para todo **x**. `twice doble x` = `cuadruple x`?
- ¿Cómo demostramos un para todo?
  - Una forma es elegir un elemento *arbitrario* del conjunto, y demostrarlo para ese elemento

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice doble` = `cuadruple`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `twice doble x` = `cuadruple x`?

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice doble` = `cuadruple`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `twice doble x` = `cuadruple x`?

  Sea **n** un número cualquiera. Se verá que `twice doble n` = `cuadruple n`

Principio de extensionalidad

> *Principio de extensionalidad*
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice doble` = `cuadruple`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `twice doble x` = `cuadruple x`?

  Sea **n** un número cualquiera. Se verá que `twice doble n` = `cuadruple n`

```
   twice doble n
=                    (twice)
   doble (doble n)
=                    (doble)
   doble (n+n)
=                    (doble)
   (n+n) + (n+n)
```

Principio de extensionalidad

> *Principio de extensionalidad*
> ¿**f** = **g**?
> es equivalente a
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice doble = cuadruple`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `twice doble x = cuadruple x`?

  Sea **n** un número cualquiera. Se verá que `twice doble n = cuadruple n`

	`twice doble n`			`cuadruple n`
=		(`twice`)	=	(`cuadruple`)
	`doble (doble n)`			`4*n`
=		(`doble`)	=	(aritm.)
	`doble (n+n)`			`(n+n)+(n+n)`
=		(`doble`)		
	`(n+n)+(n+n)`			

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice doble` = `cuadruple`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `twice doble x` = `cuadruple x`?

  Sea **n** un número cualquiera. Se verá que `twice doble n` = `cuadruple n`

```
  twice doble n
=                     (twice)
  doble (doble n)
=                     (doble)
  doble (n+n)
=                     (doble)
  (n+n)+(n+n)
```

```
  cuadruple n
=                     (cuadruple)
  4*n
=                     (aritm.)
  (n+n)+(n+n)
```

Ambos lados llegan a lo mismo.

Vale la propiedad

Principio de extensionalidad

- Prop.: ¿`twice twice = \f x -> f (f (f (f x)))`?
  Dem.:

Principio de extensionalidad

> *Principio de extensionalidad*
>
> ¿f = g?
>
> es equivalente a
>
> ¿para todo **x**. f x = g x?

- **Prop.:** ¿$\underbrace{\boxed{\texttt{twice twice}}}^{f}$ = $\overbrace{\boxed{\texttt{\textbackslash f x -> f (f (f (f x)))}}}^{g}$?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo $\overset{x}{\boxed{\text{f}}}$.

  $\underbrace{\boxed{\texttt{twice twice}}}_{f}\ \underbrace{\boxed{\text{f}}}_{x}$ = $\underbrace{\boxed{\texttt{(\textbackslash f x -> f (f (f (f x))))}}}_{g}\ \underbrace{\boxed{\text{f}}}_{x}$?

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **f** .

  `twice twice` **f** = `(\f x -> f (f (f (f x))))` **f** ?

  Como aún son funciones, se aplica nuevamente el ppio. de ext.

Principio de extensionalidad

> *Principio de extensionalidad*
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **f** .

  $\underbrace{\boxed{\texttt{twice twice f}}}_{f'}$ = $\underbrace{\boxed{\texttt{(\textbackslash f x -> f (f (f (f x)))) f}}}_{g}$?

  Como aún son funciones, se aplica nuevamente el ppio. de ext.

  ¿para todo **f** . para todo **x** .

  $\underbrace{\boxed{\texttt{twice twice f}}}_{f'}$ x = $\underbrace{\boxed{\texttt{(\textbackslash f x -> f (f (f (f x)))) f}}}_{g}$ x ?

Principio de extensionalidad

> ***Principio de extensionalidad***
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **f** .

  `twice twice f` = `(\f x -> f (f (f (f x)))) f`?

  Como aún son funciones, se aplica nuevamente el ppio. de ext.

  ¿para todo **f** . para todo **x** .

  `twice twice f x` = `(\f x -> f (f (f (f x)))) f x`?

  Sea **h** una función y **e** una expresión.

  Se verá que `twice twice h e` = `(\f x -> f (f (f (f x)))) h e`

Principio de extensionalidad

- **Prop.:** ¿`twice twice = \f x -> f (f (f (f x)))`?

  Dem.: Se verá que `twice twice h e = (\f x -> f (f (f (f x)))) h e`

Principio de extensionalidad

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

  Dem.: Se verá que `twice twice h e` = `(\f x -> f (f (f (f x)))) h e`

```
  twice twice h e
=                (twice -- f=twice, x=h)
  twice (twice h) e
=                (twice -- f=twice h, x=e)
  twice h (twice h e)
=                (twice -- f=h, x=twice h e)
  h (h (twice h e))
=                (twice -- f=h, x=e)
  h (h (h (h e)))
```

Principio de extensionalidad

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

  Dem.: Se verá que `twice twice h e` = `(\f x -> f (f (f (f x)))) h e`

```
  twice twice h e
=                 (twice -- f=twice, x=h)
  twice (twice h) e
=                 (twice -- f=twice h, x=e)
  twice h (twice h e)
=                 (twice -- f=h, x=twice h e)
  h (h (twice h e))
=                 (twice -- f=h, x=e)
  h (h (h (h e)))
```

Principio de extensionalidad

- **Prop.:** ¿`twice twice = \f x -> f (f (f (f x)))`?

  Dem.: Se verá que `twice twice h e = (\f x -> f (f (f (f x)))) h e`

```
  twice twice h e
=                 (twice -- f=twice, x=h)
  twice (twice h) e
=                 (twice -- f=twice h, x=e)
  twice h (twice h e)
=                 (twice -- f=h, x=twice h e)
  h (h (twice h e))
=                 (twice -- f=h, x=e)
  h (h (h (h e)))
```

Principio de extensionalidad

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

Dem.: Se verá que `twice twice h e` = `(\f x -> f (f (f (f x)))) h e`

```
  twice twice h e
=                (twice -- f=twice, x=h)
  twice (twice h) e
=                (twice -- f=twice h, x=e)
  twice h (twice h e)
=                (twice -- f=h, x=twice h e)
  h (h (twice h e))
=                (twice -- f=h, x=e)
  h (h (h (h e)))
```

```
  (\f x -> f (f (f (f x)))) h e
=                (regla beta, f=h)
  (\x -> h (h (h (h x)))) e
=                (regla beta. x=e)
  h (h (h (h e)))
```

Principio de extensionalidad

- **Prop.:** ¿`twice twice` = `\f x -> f (f (f (f x)))`?

Dem.: Se verá que `twice twice h e` = `(\f x -> f (f (f (f x)))) h e`

```
  twice twice h e
=                  (twice -- f=twice, x=h)
  twice (twice h) e
=                  (twice -- f=twice h, x=e)
  twice h (twice h e)
=                  (twice -- f=h, x=twice h e)
  h (h (twice h e))
=                  (twice -- f=h, x=e)
  h (h (h (h e)))
```

```
  (\f x -> f (f (f (f x)))) h e
=                  (regla beta, f=h)
  (\x -> h (h (h (h x)))) e
=                  (regla beta. x=e)
  h (h (h (h e)))
```

Ambos lados llegan a lo mismo.

Vale la propiedad

Principio de extensionalidad

Principio de extensionalidad

¿**f = g**?

es equivalente a

¿para todo **x**. **f x = g x**?

- Prop.: ¿`curry suma' = suma`?

  Dem.:

Principio de extensionalidad

> *Principio de extensionalidad*
>
> ¿**f** = **g**?
>
> es equivalente a
>
> ¿para todo **x**. **f x** = **g x**?

- **Prop.:** ¿`curry suma'` = `suma`?

  Dem.: Por el ppio. de ext. (dos veces), es equivalente demostrar que

  ¿para todo **x**. `curry suma' x` = `suma x`?

  ¿para todo **x**. para todo **y**. `curry suma' x y` = `suma x y`?

Principio de extensionalidad

- **Prop.:** ¿`curry suma' = suma`?

  Dem.: Por el ppio. de ext. (dos veces), es equivalente demostrar que

  ¿para todo **x**. `curry suma' x = suma x`?

  ¿para todo **x**. para todo **y**. `curry suma' x y = suma x y`?

  Sean **n** y **m** dos números. Se verá que `curry suma' n m = suma n m`

Principio de extensionalidad

- **Prop.:** ¿`curry suma' = suma`?

  Dem.: Por el ppio. de ext. (dos veces), es equivalente demostrar que

  ¿para todo **x**. `curry suma' x = suma x`?

  ¿para todo **x**. para todo **y**. `curry suma' x y = suma x y`?

  Sean **n** y **m** dos números. Se verá que `curry suma' n m = suma n m`

```
    curry suma' n m
=                     (curry -- f=suma', x=n, y=m)
    suma' (n,m)
=                     (suma', x=n, y=m)
    n+m
```

Principio de extensionalidad

- **Prop.:** ¿`curry suma' = suma`?

  Dem.: Por el ppio. de ext. (dos veces), es equivalente demostrar que

  ¿para todo **x**. `curry suma' x = suma x`?

  ¿para todo **x**. para todo **y**. `curry suma' x y = suma x y`?

  Sean **n** y **m** dos números. Se verá que `curry suma' n m = suma n m`

`curry suma' n m`	`suma n m`
= (`curry` -- `f=suma'`, `x`=n, `y`=m)	= (`suma`, `x`=n, `y`=m)
`suma' (n,m)`	n+m
= (`suma'`, `x`=n, `y`=m)	
n+m	

Principio de extensionalidad

- **Prop.:** ¿`curry suma' = suma`?

  Dem.: Por el ppio. de ext. (dos veces), es equivalente demostrar que

  ¿para todo **x**. `curry suma' x = suma x`?

  ¿para todo **x**. para todo **y**. `curry suma' x y = suma x y`?

  Sean **n** y **m** dos números. Se verá que `curry suma' n m = suma n m`

`curry suma' n m`	`suma n m`
= (`curry` -- `f=suma'`, `x=n`, `y=m`)	= (`suma`, `x=n`, `y=m`)
`suma' (n,m)`	n+m
= (`suma'`, `x=n`, `y=m`)	
n+m	

Ambos lados llegan a lo mismo.
Vale la propiedad

Motivación del principio de inducción

Problemas con ciertas ecuaciones

- Consideremos las siguientes definiciones

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact (n-1)
```

```
factL :: Int -> Int
factL 0 = 1
factL n = factL (n-1) * n
```

  - ¿Cuál es la relación entre ambas funciones?
    - ¿Podemos demostrar que **`fact`** = **`factL`**? ¿Cómo?

Problemas con ciertas ecuaciones

- Prop.: ¿`fact = factL`?
  Dem.:

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `fact x = factL x`?

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `fact x = factL x`?

  Sea **n** un número. Se verá que `fact n = factL n`

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `fact x` = `factL x`?

  Sea **n** un número. Se verá que `fact n` = `factL n`

  - La definición de `fact` está separada en casos
  - Deben considerarse diferentes posibilidades para **n**

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

  Dem.: Por el ppio. de ext., es equivalente demostrar que

  ¿para todo **x**. `fact x` = `factL x`?

  Sea **n** un número. Se verá que `fact n` = `factL n`

  - La definición de `fact` está separada en casos
  - Deben considerarse diferentes posibilidades para **n**
  - *Análisis de casos*
    - Caso **n** = 0
    - Caso **n** ≠ 0

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Se verá que `fact n = factL n` por análisis de casos

  Caso **n** = 0)

  Caso **n** ≠ 0)

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Se verá que `fact n = factL n` por análisis de casos

  Caso `n = 0`)

  ¿`fact 0 = factL 0`?

  Caso `n ≠ 0`)

  ¿`fact n = factL n`?

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Se verá que `fact n = factL n` por análisis de casos

  Caso n = 0)

  ¿`fact 0 = factL 0`?

  ```
    fact 0
  =                (fact.1)
    1
  ```

  Caso n ≠ 0)

  ¿`fact n = factL n`?

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

  Dem.: Se verá que `fact n` = `factL n` por análisis de casos

Caso **n** = 0)

¿`fact 0` = `factL 0`?

<u>`fact 0`</u>		<u>`factL 0`</u>		
=	(`fact.1`)	=	(`factL.1`)	
`1`		`1`		Vale este caso

Caso **n** ≠ 0)

¿`fact n` = `factL n`?

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

  Dem.: Se verá que `fact n = factL n` por análisis de casos

Caso n = 0)
¿`fact 0 = factL 0`?

```
   fact 0                          |      factL 0
=                      (fact.1)    |   =               (factL.1)
   1                               |      1
```

Vale este caso

Caso n ≠ 0)
¿`fact n = factL n`?

```
   fact n
=                      (fact.2 -- n=n)
   n * fact (n-1)
=                      (aritm.)
   fact (n-1) * n
```

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact = factL`?

Dem.: Se verá que `fact n = factL n` por análisis de casos

Caso n = 0)

¿`fact 0 = factL 0`?

```
   fact 0                          |      factL 0
=                     (fact.1)     |   =                 (factL.1)
   1                               |      1
```

Vale este caso

Caso n ≠ 0)

¿`fact n = factL n`?

```
   fact n                          |      factL n
=                (fact.2 -- n=n)   |   =                 (factL.2, n=n)
   n * fact (n-1)                  |      factL (n-1) * n
=                     (aritm.)     |
   fact (n-1) * n
```

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

Dem.: Se verá que `fact n` = `factL n` por análisis de casos

Caso **n** = 0)

¿`fact 0` = `factL 0`?

```
  fact 0                        |    factL 0
=            (fact.1)           |  =            (factL.1)
  1                             |    1
```

Vale este caso

Caso **n** ≠ 0)

¿`fact n` = `factL n`?

```
  fact n                        |    factL n
=            (fact.2 -- n=n)    |  =            (factL.2, n=n)
  n * fact (n-1)                |    factL (n-1) * n
=            (aritm.)           |
  fact (n-1) * n
```

¿Qué se requiere para que sean iguales?

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

  Dem.: Se verá que `fact n` = `factL n` por análisis de casos

Caso **n** = 0)

¿`fact 0` = `factL 0`?

```
   fact 0
=                       (fact.1)
   1
```

```
   factL 0
=                       (factL.1)
   1
```

Vale este caso

Caso **n** ≠ 0)

¿`fact n` = `factL n`?

```
   fact n
=                       (fact.2 -- n=n)
   n * fact (n-1)
=                       (aritm.)
   fact (n-1) * n
```

```
   factL n
=                       (factL.2, n=n)
   factL (n-1) * n
```

¿Qué se requiere para que sean iguales?

¡Lo mismo que queremos demostrar!

Problemas con ciertas ecuaciones

- **Prop.:** ¿`fact` = `factL`?

  Dem.: Se verá que `fact n` = `factL n` por análisis de casos

Caso n = 0)

¿`fact 0` = `factL 0`?

```
   fact 0                          |     factL 0
=                  (fact.1)        |  =                  (factL.1)
   1                               |     1
```

Vale este caso

Caso n ≠ 0)

¿`fact n` = `factL n`?

```
   fact n                          |     factL n
=                  (fact.2 -- n=n) |  =                  (factL.2, n=n)
   n * fact (n-1)                  |     factL (n-1) * n
=                  (aritm.)
   fact (n-1) * n
```

¿Qué se requiere para que sean iguales?

¡Lo mismo que queremos demostrar!

- Es necesaria alguna herramienta más…

Problemas con ciertas ecuaciones

- Volvamos a la definición de fact

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact (n-1)
```

  - ¿Cuánto vale **`fact (-1)`**?

Problemas con ciertas ecuaciones

- Volvamos a la definición de fact

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact (n-1)
```

  - ¿Cuánto vale `fact (-1)`?
    - `fact (-1)` = ⊥

Problemas con ciertas ecuaciones

- Volvamos a la definición de fact

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact (n-1)
```

  - ¿Cuánto vale `fact (-1)`?
    - `fact (-1)` = ⊥
- ¿Son ecuaciones orientadas?

Es similar a
`f n = f n`

Problemas con ciertas ecuaciones

- Volvamos a la definición de fact

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact (n-1)
```

  - ¿Cuánto vale `fact (-1)`?
    - `fact (-1)` = ⊥
- ¿Son ecuaciones orientadas?
  - Lo sorprendente es que funcione para **n>0**…

Es similar a
`f n = f n`

Motivación para nueva herramienta

- Problemas con ciertas ecuaciones
  - ¿Cómo saber si son ecuaciones orientadas?
    - E.g.: `fact`
  - ¿Qué falta para poder demostrar en estos casos?
    - E.g.: ¿`fact` = `factL`?
- Expresividad de los tipos algebraicos
  - ¿Cómo definir tipos algebraicos con infinitos elementos?

Motivación para nueva herramienta

- Las 3 se solucionan con la misma herramienta
  - ¿Cómo saber si son ecuaciones orientadas?
  - ¿Qué falta para poder demostrar en estos casos?
  - ¿Cómo definir tipos algebraicos con infinitos elementos?

Motivación para nueva herramienta

- Las 3 se solucionan con la misma herramienta
  - ¿Cómo saber si son ecuaciones orientadas?
  - ¿Qué falta para poder demostrar en estos casos?
  - ¿Cómo definir tipos algebraicos con infinitos elementos?
- Parece ser necesaria ***inducción*** o ***recursión***
  - ¿Realmente entendemos la inducción matemática?
  - ¿Qué es la inducción? ¿Qué es la recursión?

Resumen

Resumen

- Propiedades y demostraciones
  - Demostraciones manuales formales
    - Directas
    - Con “para todo”
    - Con análisis de casos
    - Consideración por la no terminación
  - Equivalencia de funciones
    - Principio de extensionalidad
    - Principio de inducción…