

Propiedades y demostraciones

Programación funcional

Hasta ahora vimos:

- ▶ Valores, expresiones y reducción
- ▶ Tipos, currificación y funciones de alto orden
- ▶ Recursión

¿Qué es una propiedad?

Propiedad

Una *propiedad* es una sentencia sobre un elemento que puede ser verdadera o falsa.

Ejemplos de propiedades sobre programas:

- ▶ Terminación
- ▶ Equivalencia
- ▶ Correctitud¹
- ▶ (No-)Vacuidad

¹La palabra correcta en castellano es corrección.

¿Cómo verificamos una equivalencia?

Demostración

Una *demostración* es una secuencia de transformaciones bien fundadas llegando a una conclusión válida a partir de una premisa dada.

Ejemplos:

```
True || False ≡ True
```

```
nul [] ≡ True
```

En nuestro caso usamos el mecanismo de cómputo por reducción para justificar transformaciones:

- ▶ Cada paso tiene que estar justificado por un axioma (e.g. regla de reducción) o una propiedad conocida.
- ▶ La conclusiones debe ser evidentemente verdadera (e.g. expresiones sintácticamente equivalentes).

¿Son equivalentes?

$$\lambda x \rightarrow x \equiv \lambda y \rightarrow y$$

α -equivalencia

Las variables ligadas pueden renombrarse con un nombre fresco (α -conversión) sin alterar la semántica de la expresión.

Entonces, renombrado y por x:

$$\lambda x \rightarrow x \equiv \lambda x \rightarrow x$$

Dos expresiones sintácticamente idénticas son trivialmente equivalentes.

¿Cómo determinamos que dos funciones son equivalentes?

Principio de extensionalidad

Dos funciones son equivalentes si son equivalentes **para todo** parámetro.

Ejemplos:

```
curry (uncurry f)  $\equiv$  f
```

```
flip (curry f)  $\equiv$  curry (f . swap)
```

Un cuantificador permite especificar la cantidad de “especímenes” en un dominio de discurso que satisfacen una fórmula abierta.

Ejemplos:

```
( $\forall$  x :: Nat)(odd x || even x  $\equiv$  True) -- todos
```

```
 $\neg$ (( $\exists$  x :: Bool)(x == not x  $\equiv$  True)) -- al menos uno
```

Separación en casos

¿Cómo demostramos que

$(\forall b :: \text{Bool})(\text{if } b \text{ then True else False} \equiv \text{id } b)$

Separando en casos:

1. $b = \text{True}$
2. $b = \text{False}$

Para que la demostración sea correcta la separación en casos debe:

- ▶ ser exhaustiva; y
- ▶ llegar a una conclusión válida en cada caso.

Correspondencia Curry-Howard

Un programa es una prueba para la fórmula que representa su tipo.



William Howard

Ejemplo: Si existen funciones totales con los siguientes tipos, entonces las fórmulas que representan los tipos son verdaderas.

- ▶ $f :: a \rightarrow a$
- ▶ $g :: a \rightarrow b$

En sistemas de tipos más complejos es posible expresar propiedades más ricas.