Automatización en el Gallo Mecánico

Beta Ziliani

Max Planck Institute for Software Systems (MPI-SWS)

trabajo conjunto con Georges Gonthier (MSRC),
Aleks Nanevski (IMDEA), y Derek Dreyer (MPI-SWS)

Drunk talks, Octubre, 2011, UBA-FCEyN

El demostrador de teoremas Coq

Theorem suma_conmuta : $\forall n\ m : \mathbb{N}, n + m = m + n$.

Proof.

intro n.

induction n.

...

Qed.

El demostrador de teoremas Coq

Theorem suma_conmuta : $\forall n\ m : \mathbb{N}, n + m = m + n$.

Proof.

intro n.

induction n.

...

Qed.

El demostrador de teoremas Coq

Theorem suma_conmuta : $\forall n\ m : \mathbb{N}, n + m = m + n$.

Proof.

intro n.

induction n.

...

Qed.

No sólo demuestra ejemplos sencillos

- Teorema de los cuatro colores (Gonthier et al.)
- CompCert: compilador certificado para C (Leroy et al.)
- Millones de pruebas en el area de lenguajes de programación
- Cada vez mas utilizado en el aula:
  - Lógica
  - Lenguajes de programación
  - ...
- ...

Hay otros demostradores de teoremas.

Breve introducción a listas

Lista de naturales del 1 al 5:

$$(1 :: 2 :: 3 :: 4 :: 5 :: \text{vacia})$$

Breve introducción a listas

Lista de naturales del 1 al 5:

$$(1 :: 2 :: 3 :: 4 :: 5 :: \text{vacia})$$

Predicado de pertenencia:

$$\text{en } x\ s$$

Breve introducción a listas

Lista de naturales del 1 al 5:

$$(1 :: 2 :: 3 :: 4 :: 5 :: \text{vacia})$$

Predicado de pertenencia:

$$\text{en } x \; s$$

Lemas para demostrar pertenencia:

$$\text{en\_cabeza} : \forall x \; s, \; \text{en } x \; (x :: s)$$

$$\text{en\_cola} : \forall x \; y \; s, \; \text{en } x \; s \underbrace{\longrightarrow}_{\text{implica}} \text{en } x \; (y :: s)$$

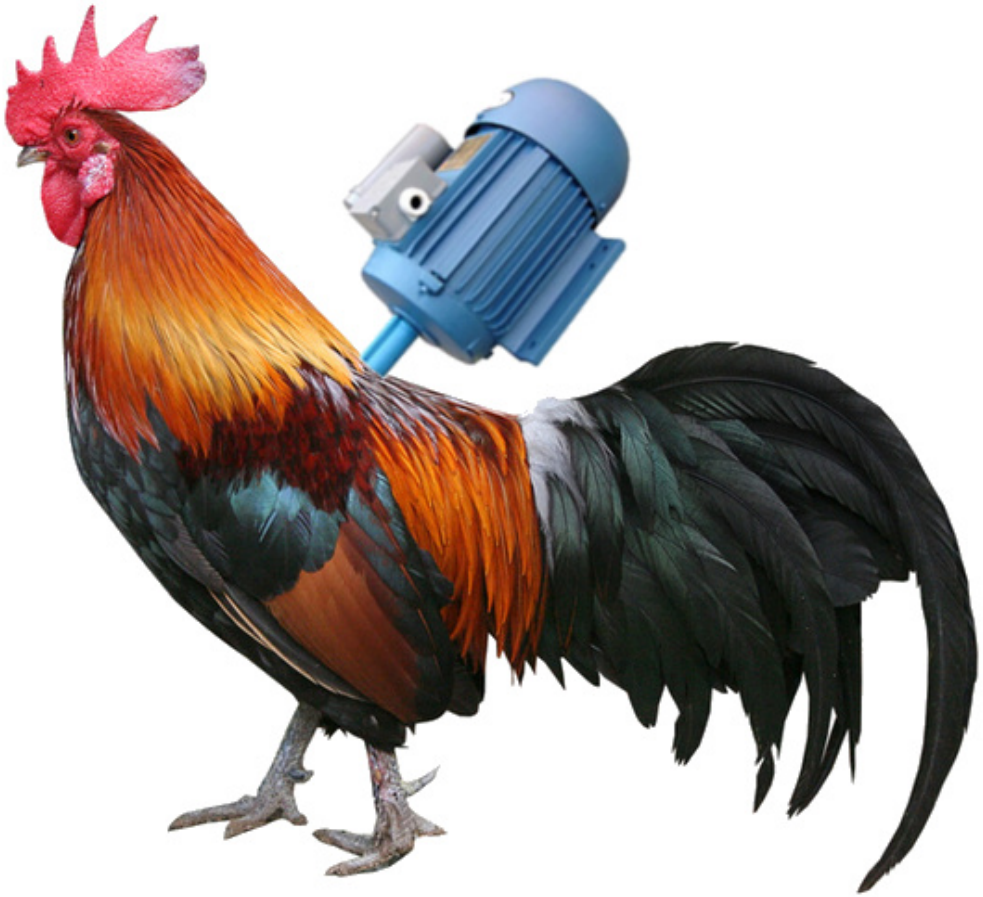
$$\text{en\_vacio} : \forall x, \underbrace{\sim}_{\text{negacion}} (\text{en } x \text{ vacia})$$

Demo

Sin automatización...

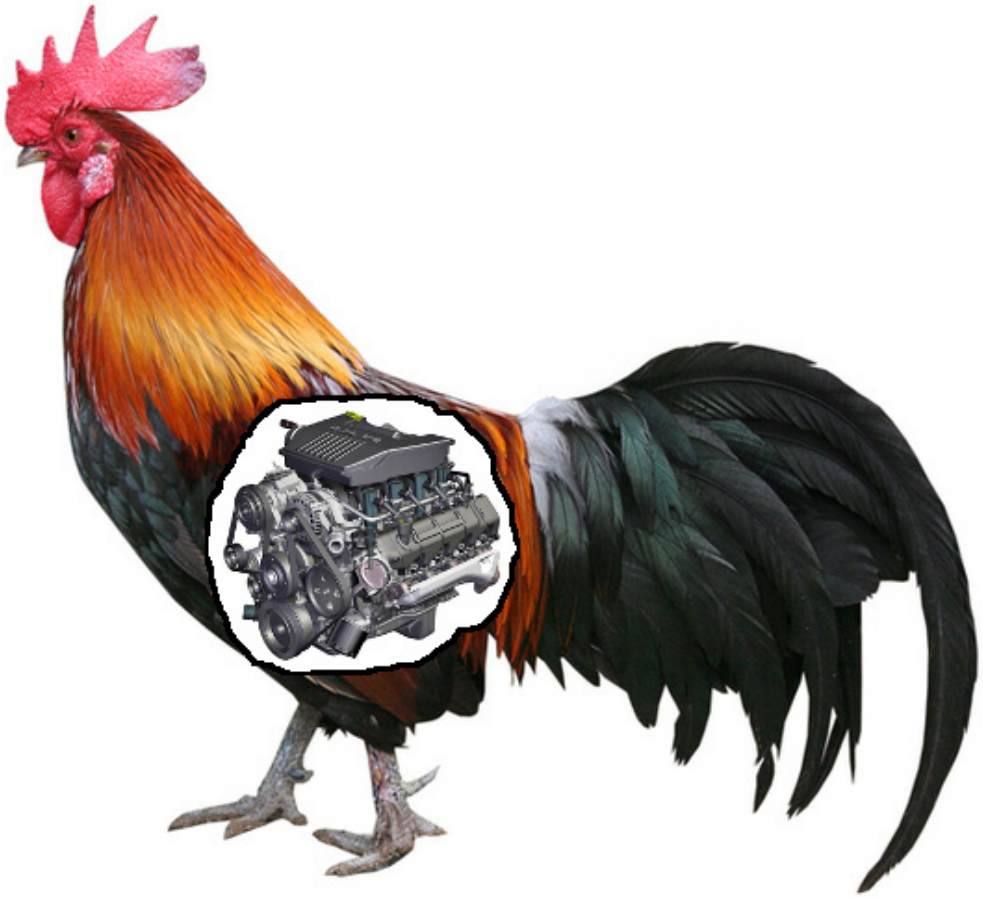
Es imposible!!!

Automatizacion con Ltac



Tiene algunos problemas...

Nuestro descubrimiento



Debajo del plumaje se esconde un V8!

Tipos

Tipos (ahora en serio)

Un tipo (type en inglés) es como un conjunto de valores.

Ejemplos:

$0, 1, 2, 3, \ldots$	tienen tipo	nat
$\text{vacia}, (1 :: \text{vacia}), \ldots$	tienen tipo	lista nat
$\text{true}, \text{false}$	tienen tipo	bool
nat	tiene tipo	Type
bool	tiene tipo	Type
lista nat	tiene tipo	Type

Typechecker (verificador de tipos)

Un typechecker verifica que no hagamos macanas.

Ejemplos:

$$0 + 1$$

$$0 + \text{vacia}$$

$$(1 :: (1 + 1) :: \text{vacia})$$

$$(1 :: \text{vacia} :: \text{vacia})$$

Typechecker (verificador de tipos)

Un typechecker verifica que no hagamos macanas.

Ejemplos:

$0 + 1$

$0 + \text{vacia}$

$(1 :: (1 + 1) :: \text{vacia})$

$(1 :: \text{vacia} :: \text{vacia})$

Typechecker (verificador de tipos)

Un typechecker verifica que no hagamos macanas.

Ejemplos:

$0 + 1$

$0 + \text{vacia}$

$(1 :: (1 + 1) :: \text{vacia})$

$(1 :: \text{vacia} :: \text{vacia})$

Typechecker (verificador de tipos)

Un typechecker verifica que no hagamos macanas.

Ejemplos:

$0 + 1$

$0 + \text{vacia}$

$(1 :: (1 + 1) :: \text{vacia})$

$(1 :: \text{vacia} :: \text{vacia})$

Typechecker (verificador de tipos)

Un typechecker verifica que no hagamos macanas.

Ejemplos:

$0 + 1$ 

$0 + \text{vacia}$ 

$(1 :: (1 + 1) :: \text{vacia})$ 

$(1 :: \text{vacia} :: \text{vacia})$ 

También el gallo computa

- Coq incluye un lenguaje funcional
- Ejemplo, función identidad:

$$\underline{\text{Definition}}\ \text{identidad} := \underline{\text{fun}}\ (T : \text{Type})\ (x : T)\ \Rightarrow x$$

- Podemos computar

$$2 + 2 = 4$$
$$\text{identidad nat } 5 = 5$$
$$\text{identidad bool true} = \text{true}$$

Cómo funciona?

```
Lemma tautologia : ∀P : Prop, P ⟶ P.
Proof.
  intro P.
  intro x.
  apply x.
Qed.
```

Basado en la correspondencia *Curry-Howard*:

- Un lema no es nada más que una función
- Una demostración no es nada más que el código
- El "gallo" es un typechecker!

Sobrecarga en Coq

Igualdad para todos y todas!

Motivación: único símbolo para representar igualdad.

Sobrecarga en Coq

Igualdad para todos y todas!

Motivación: único símbolo para representar igualdad.

Ejemplos:

$x == 1$ (naturales)

Sobrecarga en Coq

Igualdad para todos y todas!

Motivación: único símbolo para representar igualdad.

Ejemplos:

$x == 1$ (naturales)

$b == \text{true}$ (booleanos)

Sobrecarga en Coq

Igualdad para todos y todas!

Motivación: único símbolo para representar igualdad.

Ejemplos:

$x == 1$ (naturales)

$b == \text{true}$ (booleanos)

$(x, \text{true}) == (\text{false}, y)$ (par de booleanos)

Sobrecarga en Coq

Igualdad para todos y todas!

Motivación: único símbolo para representar igualdad.

Ejemplos:

$x == 1$	(naturales)
$b == \text{true}$	(booleanos)
$(x, \text{true}) == (\text{false}, y)$	(par de booleanos)
$\text{bla} == \text{ble}$	(mi tipo)

Ajá, muy linda la sobrecarga. Y a mi qué???

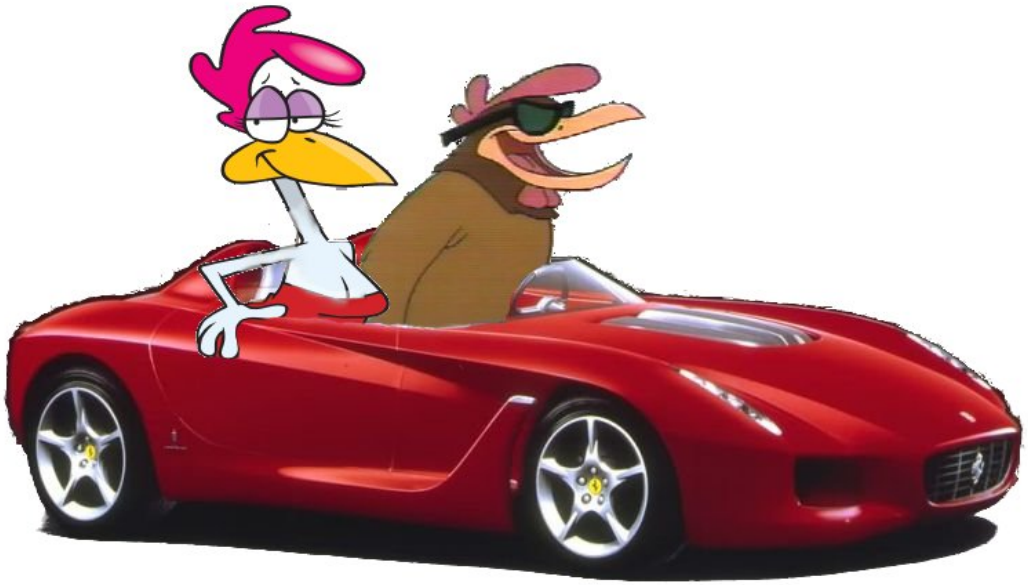
Automatizar pruebas mediante “sobrecarga de lemas”

Correspondencia Curry-Howard “sobrecargada”

- Sobrecarga:
  infiere el codigo de una función basada en sus argumentos
- Sobrecarga de lemas:
  infiere la prueba de un lema basado en sus argumentos

Rutinas de automatización robustas, verificables!

Eso es todo amigos!

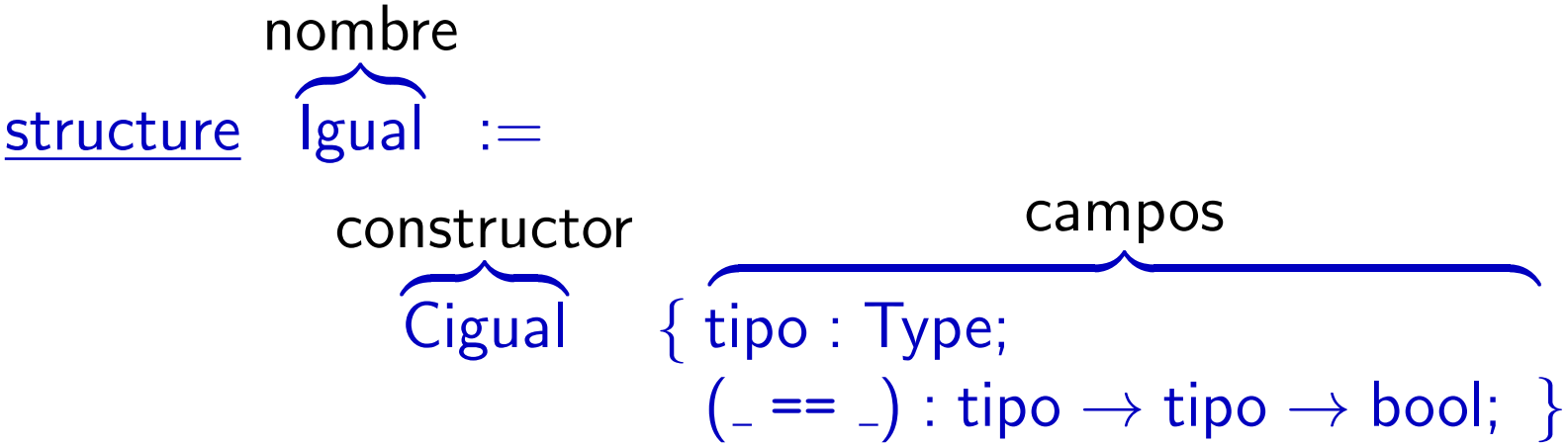


Preguntas?

Sobrecarga en Coq

Igualdad para todos y todas!

Estructura: similar a una clase (Haskell) o interfaz (Java)



Sobrecarga en Coq

Igualdad para todos y todas!

Estructura: similar a una clase (Haskell) o interfaz (Java)

$$\underline{\text{structure}}\ \overbrace{\text{Igual}}^{\text{nombre}}\ :=\ \overbrace{\text{Cigual}}^{\text{constructor}}\ \overbrace{\{\ \text{tipo} : \text{Type};\ (\_ == \_) : \text{tipo} \to \text{tipo} \to \text{bool};\ \}}^{\text{campos}}$$

Coq crea un proyector por cada campo, e.g.:

$$\begin{aligned} \text{tipo} &: \text{Igual} \to \text{Type} \\ (\_ == \_) &: \forall e : \text{Igual}.\ \text{tipo}\ e \to \text{tipo}\ e \to \text{bool} \end{aligned}$$

Instancias

Se definen instancias para cada tipo
(en este caso, sólo para booleanos y pares)

$$\underline{\text{canonical}}\ \text{bool\_inst} := \text{Cigual}\ \overbrace{\text{bool}}^{\text{tipo}}\ \overbrace{\text{igual\_bool}}^{(\_ == \_)}$$

$$\underline{\text{canonical}}\ \text{par\_inst}\ (A\ B : \text{Igual}) :=$$
$$\text{Cigual}\ \underbrace{(\text{tipo}\ A \times \text{tipo}\ B)}_{\text{tipo}}\ \underbrace{(\text{igual\_par}\ A\ B)}_{(\_ == \_)}$$

Ejemplo de inferencia

$$(x, \text{true}) == (\text{false}, y)$$

Ejemplo de inferencia

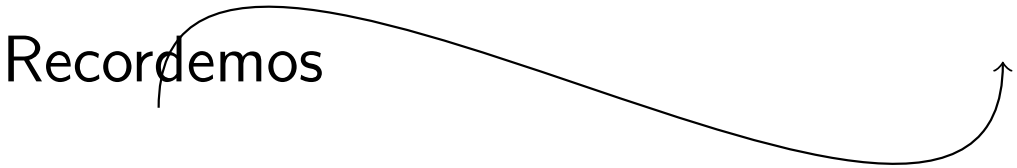
$$(x, \text{true}) == (\text{false}, y)$$

Recordemos

$$(\_ == \_) \ : \ \forall\ e : \text{Igual}\ .\ \ \text{tipo}\ e \to \text{tipo}\ e \to \text{bool}$$

Ejemplo de inferencia

$$(x, \text{true}) == (\text{false}, y)$$



$$(\_ == \_) \;:\; \forall\, e : \text{Igual} \;.\; \text{tipo}\ e \to \text{tipo}\ e \to \text{bool}$$

parámetro implícito

Ejemplo de inferencia

$$(x, \text{true}) == (\text{false}, y)$$

Recordemos

$$(\_ == \_) \ : \ \forall \ e : \text{Igual} \ . \ \text{tipo} \ e \to \text{tipo} \ e \to \text{bool}$$

Coq encuentra una instancia de $e : \text{Igual}$ tal que

$$\text{tipo} \ e \quad \text{es igual a} \quad (\text{bool} \times \text{bool})$$

Ejemplo de inferencia

$$(x, \text{true}) == (\text{false}, y)$$

Recordemos

$$(\_ == \_) \ : \ \forall \ e : \text{Igual} \ . \ \ \text{tipo} \ e \to \text{tipo} \ e \to \text{bool}$$

Coq encuentra una instancia de $e : \text{Igual}$ tal que

$$\text{tipo} \ e \quad \text{es igual a} \quad (\text{bool} \times \text{bool})$$

$$\downarrow$$

$$e = \text{par\_inst} \ \text{bool\_inst} \ \text{bool\_inst}$$