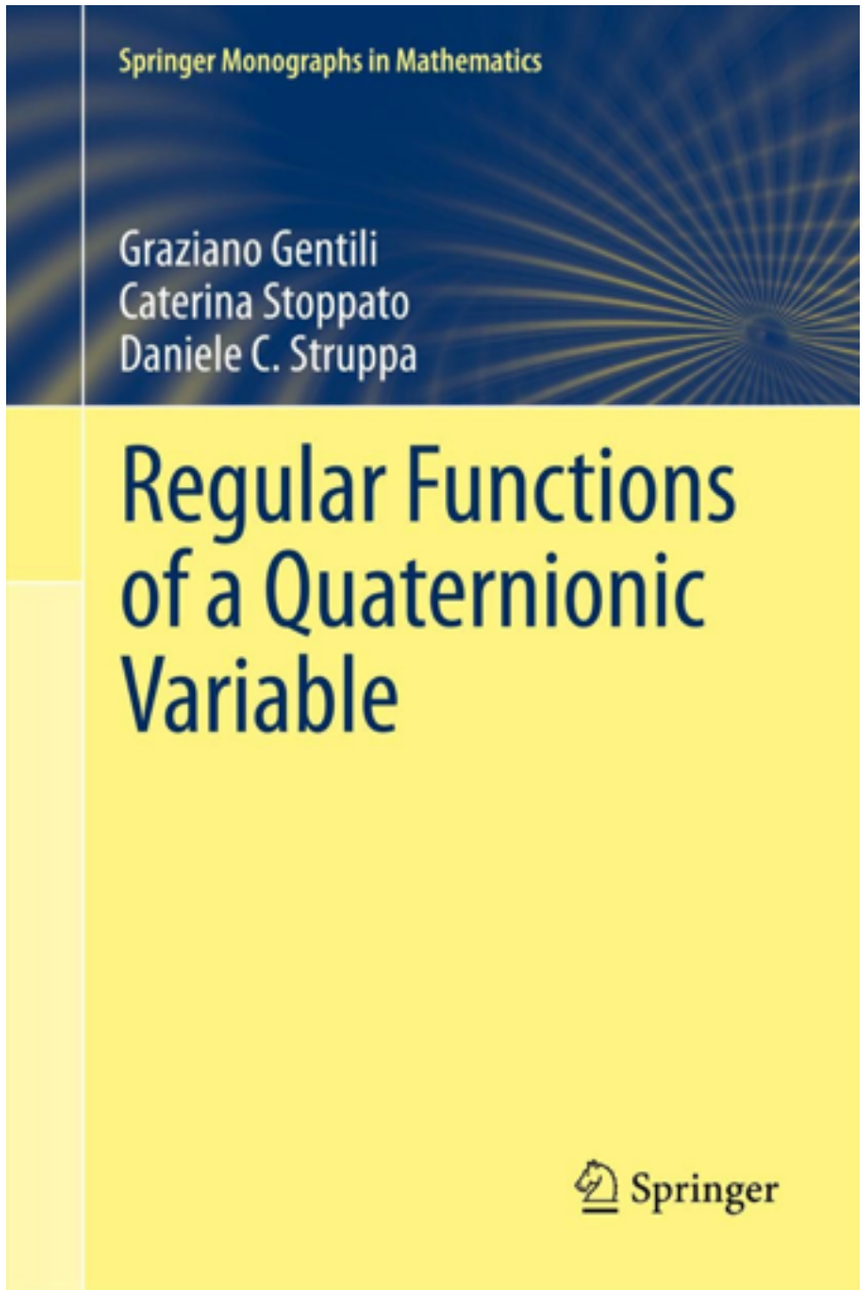
Meccanizzazione del Calcolo Quaternionico

Marco Maggesi

Incontro Nazionale di Analisi Ipercomplessa
Firenze, 23 gennaio 2015

Progetto di meccanizzazione

Meccanizzare alcuni teoremi di base delle funzioni *slice* regolari.



Cos’è la meccanizzazione?

- Formalizzazione
- Scrivere l’**enunciato** di un teorema in un linguaggio completamente formale;
- Scrivere la **dimostrazione** del teorema usando un insieme prefissato di regole e assiomi;

- Meccanizzazione
- Verificare **algoritmicamente** la dimostrazione formale usando un computer.

Nota: la **formalizzazione è un prerequisito della meccanizzazione**, quindi parleremo di meccanizzazione per indicare l’intero processo.

1		$\forall y \neg P(y)$	Assumption
2		$\exists x P(x)$	
3	u	$P(u)$	
4		$\forall y \neg P(y)$	R, 1
5		$\neg P(u)$	$\forall$E, 4
6		$\bot$	$\neg$E, 3,5
7		$\bot$	$\exists$E, 2, 3–6
8		$\neg \exists x P(x)$	$\neg$I, 2–7

FAQ sulla meccanizzazione

- **È teoricamente possibile?**

  “La matematica è un’Arte non si riduce ad un numero finito di regole”
  “È impossibile per motivi teorici (es. per il teorema di Gödel)”

- **È affidabile?**

  “Tutti i software contengono qualche errore, anche un dimostratore di teoremi”

- **È utile?**

  “A che serve un teorema meccanizzato?”
  “Perché dimostrare al computer teoremi già dimostrati?”

- **È fattibile in pratica?**

  “Richiede troppo tempo per essere realizzato”
  “Richiede troppe risorse di calcolo”

È teoricamente possibile?

- Risposta: **Sì!**
- Es: Russell & Whitehead, *Principia Matematica*

***103·5.** $\vdash . 0 \in \mathrm{N_0C}$

Dem.

$$\vdash . *101{\cdot}11{\cdot}12 . \supset \vdash . 0 \in \mathrm{NC} . \exists ! 0 .$$

$$[*103{\cdot}34] \qquad \supset \vdash . 0 \in \mathrm{N_0C} . \supset \vdash . \mathrm{Prop}$$

***103·51.** $\vdash . 1 \in \mathrm{N_0C}$

Dem.

$$\vdash . *101{\cdot}21{\cdot}241 . \supset \vdash . 1 \in \mathrm{NC} . \exists ! 1 .$$

$$[*103{\cdot}34] \qquad \supset \vdash . 1 \in \mathrm{N_0C} . \supset \vdash . \mathrm{Prop}$$

0 and 1 are the only cardinals of which the above property can be proved universally with our assumptions. If (as is possible so far as our assumptions go) the lowest type is a unit class, we shall have *in that type* (though in no other) $2 = \Lambda$, so that in that type $2 \sim \epsilon \mathrm{N_0C}$.

È affidabile?

"Tutti i software contengono qualche errore, anche un dimostratore di teoremi"

HOL Light è un dimostratore particolarmente semplice

- 10 regole di inferenza
- 3 assiomi
- 430 righe di codice

$$\frac{a}{\vdash a = a}$$

$$\frac{\Gamma \vdash a = b;\ \Delta \vdash b' = c}{\Gamma \cup \Delta \vdash a = c}$$

$$\frac{\Gamma \vdash f = g;\ \Delta \vdash a = b}{\Gamma \cup \Delta \vdash f\,a = g\,b}$$

$$\frac{x;\ \Gamma \vdash a = b}{\Gamma \vdash \lambda x.\ a\ = \lambda x.\ b}\ \text{(if } x \text{ is not free in } \Gamma\text{)}$$

$$\frac{(\lambda x.\ a)\,x}{\vdash (\lambda x.\ a)\,x = a}$$

$$\frac{p{:}bool}{p \vdash p}$$

$$\frac{\Gamma \vdash p;\ \Delta \vdash p' = q}{\Gamma \cup \Delta \vdash q}$$

$$\frac{\Gamma \vdash p;\ \Delta \vdash q}{(\Gamma \setminus q) \cup (\Delta \setminus p) \vdash p = q}$$

È fattibile in pratica?

Ad oggi è stata realizzata la meccanizzazione di:

- Analisi reale e complessa, algebra lineare, …
- Logica, combinatorica, computer science, …
- Verifica di hardware, software, protocolli, …
- Teorema di Feit-Thompson
- Teorema dei 4 colori
- Teorema di Hales (congettura di Keplero)

Cronologia

1995 J. Harrison, *“HOL Done Right”* (Nasce HOL Light)

2005 J. Harrison, Meccanizzazione degli spazi euclidei

2006 G. Gentili, D. Struppa, *“Cullen-regular functions”*

2007 J. Harrison, *“Formalising basic complex analysis”*

2011 Ciolli, Gentili, Maggesi, Meccanizzazione dei teoremi di Cartan I e II

2014 Maggesi, Implementazione dei quaternioni in HOL Light

Primi risultati

```
(* ========================================================== *)
(* Quaternionic calculus.                                     *)
(* File: make.hl                                              *)
(*                                                            *)
(* Copyright (c) 2014 Marco Maggesi                           *)
(* ========================================================== *)

loadt "misc.hl";;         (* Miscellanea                      *)
loadt "quaternion.hl";;   (* Basic definitions                *)
loadt "qcalc.hl";;        (* Computing with literal quaternions *)
loadt "qnormalizer.hl";; (* Normalization of quat. polynomials *)
loadt "qanal.hl";;        (* Quaternionic analysis            *)
loadt "qderivative.hl";; (* Differential of quat. functions   *)
loadt "qisom.hl";;        (* Space isometries via quaternions  *)
```

Calcolare con i quaternioni

$$\left(1 + 2\mathbf{i} - \frac{1}{2}\mathbf{k}\right)^3 = -\frac{47}{4} - \frac{5}{2}\mathbf{i} + \frac{5}{8}\mathbf{k}$$

```
# RATIONAL_QUAT_CONV
  `(Hx(&1) + Hx(&2)*ii - Hx(&1 / &2)*kk) pow 3`;;
val it : thm =
  |- (Hx(&1) + Hx(&2)*ii - Hx(&1 / &2)*kk) pow 3 =
     -- Hx(&47 / &4) - Hx(&5 / &2)*ii +
     Hx(&5 / &8)*kk
```

Polinomi sui quaternioni

$$(p+q)^3 = p^3 + q^3 + pq^2 + p^2q + pqp + qp^2 + qpq + q^2p$$

```
# QUAT_POLY_CONV ‘(x + y) pow 3‘;;
val it : thm =
  |- (p + q) pow 3 =
     p pow 3 + q pow 3 + p * q pow 2 + p pow 2 * q +
     p * q * p + q * p pow 2 + q * p * q + q pow 2 * p
```

Notazione funzionale e derivate parziali

[Spivak, *Calculus on Manifolds*]

The mere statement of [the chain rule] in classical notation requires the introduction of irrelevant letters. The usual evaluation for $D_1(f \circ (g,h))$ runs as follows:

If $f(u,v)$ is a function and $u = g(x,y)$ and $v = h(x,y)$ then

$$\frac{\partial f(g(x,y),h(x,y))}{\partial x} = \frac{\partial f(u,v)}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f(u,v)}{\partial v}\frac{\partial v}{\partial x}$$

[The symbol $\partial u/\partial x$ means $\partial/\partial x \; g(x,y)$, and $\partial/\partial u \; f(u,v)$ means $D_1 f(u,v) = D_1 f(g(x,y),h(x,y))$.] This equation is often written simply

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial x}.$$

Note that f means something different on the two sides of the equation!

Definizione di derivata *slice*

```
let has_slice_derivative = new_definition
  `!f (f':quat) net.
   (f has_slice_derivative f') net <=>
   (!l. subspace l /\ dim l = 2 /\
        Hx(&1) IN l /\ netlimit net IN l
        ==> (f has_derivative (\q. q * f')) (net within l))`;;
```

Esempio: derivata *slice* della somma

```
let HAS_SLICE_DERIVATIVE_ADD = prove
  (`!net f g.
      (f has_slice_derivative f') net /\
      (g has_slice_derivative g') net
    ==> ((\q. f q + g q) has_slice_derivative f' + g') net`,
REWRITE_TAC[has_slice_derivative] THEN
INTRO_TAC "!net f g; f g; !l; hp" THEN
USE_THEN "hp" (HYP_TAC "f" o C MATCH_MP) THEN
USE_THEN "hp" (HYP_TAC "g" o C MATCH_MP) THEN
HYP (MP_TAC o MATCH_MP HAS_DERIVATIVE_ADD o
     end_itlist CONJ) "f g" [] THEN
REWRITE_TAC[QUAT_ADD_LDISTRIB];;
```

Equivalenza con la definizione classica

```
let HAS_SLICE_DERIVATIVE = prove
 (`!f f' i x y.
     i pow 2 = -- Hx(&1) /\ ~(y = &0) /\ f differentiable at (Hx x + Hx y * i)
     ==>
    ((f has_slice_derivative f') (at (Hx x + Hx y * i)) <=>
     (?fx fy.
       ((\a. f(Hx(drop a) + Hx y*i)) has_vector_derivative fx)
          (at(lift x)) /\
       ((\b. f(Hx x + Hx(drop b)*i)) has_vector_derivative fy)
          (at(lift y)) /\
       fx + i * fy = Hx(&0) /\
       f' = fx /\
       f' = --(i * fy)))`,
[... ]);;
```

Prodotto star

```
let quat_star = new_definition
  `(f * g) (q:quat) : quat =
   if f q = Hx(&0)
   then Hx(&0)
   else f q * g (inv(f(q)) * q * f q)`;;


[...]


prove
 (`!a b. (\q. q - a) * (\q. q - b) =
         (\q:quat. q pow 2 - q * (a + b) + a * b)`,
 [...]);;
```