

Le Coq réel: comment l'apprivoiser.

Micaela Mayero

<http://www-lipn.univ-paris13.fr/~mayero/>
Université Paris 13
LIPN-LCR

GDR IM, Montpellier, 25 janvier 2007

À venir

La problématique des réels

Quels réels ?

Axiomatisation classique : pourquoi ?

Organisation de le bibliothèque

Automatisation

Existantes

Utilisateurs

Petits exemples

Exemple vrai

Conclusion

Réels vs flottants

Langages de programmation : les flottants.

Mais où est le problème ? Pourquoi ne suffisent-ils pas ici ?

Preuves mathématiques \leftarrow bonnes propriétés (\neq calcul) :

Réels vs flottants

Langages de programmation : les flottants.

Mais où est le problème ? Pourquoi ne suffisent-ils pas ici ?

Preuves mathématiques \leftarrow bonnes **propriétés** (\neq calcul) :

- ▶ l'addition dans les flottants n'est pas associative :

$$(1003 + -1000) + 7.501 = 10.50100000000000012$$

$$1003 + (-1000 + 7.501) = 10.500999999999999764$$

- ▶ successeur d'un flottant \rightarrow pas de raisonnement par dichotomie
- ▶ ...

Méfiez-vous des imitations! :-)

Quels choix avons-nous ?

- ▶ Construction
 - ▶ Cauchy-Cantor
 - ▶ coupures de Dedekind
- ▶ Axiomatisation
 - ▶ au premier ordre
 - ▶ au second ordre
- ▶ Classiques ou constructifs (intuitionnistes)

Et chez les autres prouveurs ?

- ▶ HOL; HOL-light : [construction classique](#) ; méthode de Cantor ; J.Harrison (95)
- ▶ Lego : [axiomatisation classique](#) ; C. Jones (91)
- ▶ Isabelle : [analyse non-standard](#) ; J.Fleuriot
- ▶ PVS : [axiomatisations classique + codage primitif](#) ; B.Dutertre (96), H.Gottliebsen (2000)
- ▶ Mizar : [construction classique](#) ; coupures de Dedekind + sous-ensembles
- ▶ Nuprl : [construction intuitionniste](#) ; formalisme de Bishop ; D.J.Howe (85)
- ▶ ACL2 : [analyse non standard](#) ; R.Gamboia

Dans Coq

- ▶ **axiomatisation constructive** ; M.Niqui (2000)
- ▶ **construction constructive** ; A.Ciaffaglione, P.Di Giannantonio (2001)
- ▶ **réels exacts** ; Y.Bertot
- ▶ **bibliothèque standard** : **axiomatisation classique** ; M.Mayero (98), O.Desmettre (2001)
- ▶ ...

Bibliothèque standard : axiomatisation classique

- ▶ Avantages :
 - ▶ simple à la compréhension et à l'utilisation
 - ▶ théorèmes de l'analyse
 - ▶ mathématiques classiques
- ▶ Inconvénients :
 - ▶ système intuitionniste
 - ▶ éventuelle extraction

theories/Reals

Les réels vus comme un
corps commutatif ordonné archimédien complet.

6 sous-parties :

- ▶ Axiomes et lemmes de base : commu, assoc, ordre, injection, archi, complet, ...
(Rbase : Rdefinitions, Raxioms, RIneq)
- ▶ Quelques fonctions : puissance, fact, abs, min, max, ...
(Rfunctions : R_Ifp, Rbasic_fun, R_sqr)
- ▶ Suites et séries : séries entières, suites de Cauchy, critère de d'Alembert, ...
(SeqSeries : Rseries, SeqProp, Rcomplete, Rsigma, Rprod, Cauchy_prod, Alembert)

- ▶ Fonctions trigonométriques : `cos`, ...
(`Rtrigo` : `Rtrigo_def`, `Rtrigo_fun`, `Rtrigo_alt`,
`Cos_rel`, `Cos_plus`)
- ▶ Analyse : limites, dérivées, continuité, topologie, géométrie,
coef binomiaux, th. val. inter., exp...
(`Ranalysis`, `Rtopology`, `Rsqrt`, `Rgeom`, `MVT`,
`Exp_prop`, `Rlimit`, `Rderiv`, ...)
- ▶ Intégration : Riemann, Newton, ...
(`Integration` : `RiemannInt_SF`, `RiemannInt`,
`NewtonInt`)

Automatisation : un besoin ?

Non, une nécessité ! pourquoi ? : pas de calculs.

Constantes réelles : $n = f(n)$ avec

$$f(n) = \begin{cases} 1 + 2 * f(\frac{n-1}{2}) & \text{si } n \text{ impair} \\ 2 * f(\frac{n}{2}) & \text{si } n \text{ pair} \end{cases}$$

```
Coq < Goal 2 + 1 = 3.
```

```
1 subgoal
```

```
=====
```

```
2 + 1 = 3
```

```
Unnamed_thm < info auto with real.
```

```
== apply sym_eq; apply Rplus_comm.
```

```
Proof completed.
```

Tactiques élémentaires

DiscrR, prove_sup0, prove_sup, Rcompute, split_Rmult,
split_Rabs, Reg, ...

```
Coq < Goal 2 + 1 < 4.
```

```
1 subgoal
```

```
=====
```

```
2 + 1 < 4
```

```
Unnamed_thm < prove_sup.
```

```
Proof completed.
```

```
Coq < Goal 8 + 5 <> 0.
```

```
1 subgoal
```

```
=====
```

```
8 + 5 <> 0
```

```
Unnamed_thm < discrR.
```

```
Proof completed.
```

Utilisation de \mathcal{L}_{tac}

Exercice : que fais-je ?

```
Ltac que_fais_je:=  
  match goal with  
  | |- (0 < 1) => apply Rlt_0_1  
  | |- (0 < ?X1) =>  
    repeat  
      (apply Rmult_lt_0_compat || apply Rplus_lt_pos;  
        try apply Rlt_0_1 || apply Rlt_R0_R2)  
  | |- (?X1 > 0) => change (0 < X1) in |- *; que_fais_je  
end.
```

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

```
Ltac toreplace_gt :=  
match goal with
```

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

```
Ltac toreplace_gt :=  
match goal with  
| H : (?X1 > ?X2) |- _ =>
```

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

```
Ltac toreplace_gt :=  
match goal with  
| H : (?X1 > ?X2) |- _ => change (X2 < X1) in H
```

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

```
Ltac toreplace_gt :=  
match goal with  
| H : (?X1 > ?X2) |- _ => change (X2 < X1) in H  
| |- (?X1 > ?X2) => change (X2 < X1)  
end.
```

Exercice d'automatisation

Écrire une tactique qui remplace tous les $>$ pas des $<$ (hyp et goal).
Open Local Scope R_scope.

```
Ltac toreplace_gt :=  
match goal with  
| H : (?X1 > ?X2) |- _ => change (X2 < X1) in H  
| |- (?X1 > ?X2) => change (X2 < X1)  
end.
```

```
Ltac replace_gtR := repeat toreplace_gtR.
```

Application

```
Coq < Goal forall a b c : R, a > b -> b > c -> a > c.
```

```
1 subgoal
```

```
=====
```

```
forall a b c : R, a > b -> b > c -> a > c
```

```
Unnamed_thm < intros;replace_gtR.
```

```
1 subgoal
```

```
a : R
```

```
b : R
```

```
c : R
```

```
H : b < a
```

```
H0 : c < b
```

```
=====
```

```
c < a
```

$$(f + g)' = f' + g'$$

Lemma Dadd :

```
forall (D:R -> Prop) (df dg f g:R -> R) (x0:R),
  D_in f df D x0 ->
  D_in g dg D x0 ->
  D_in (fun x:R => f x + g x) (fun x:R => df x + dg x) D x0.
```

Proof.

```
unfold D_in in |- *; intros;
generalize
  (limit_plus (fun x:R => (f x - f x0) * / (x - x0))
    (fun x:R => (g x - g x0) * / (x - x0)) (D_x D x0)
    (df x0) (dg x0) x0 H H0); clear H H0; unfold limit1_in in |- *;
unfold limit_in in |- *; simpl in |- *; intros; elim (H eps H0);
clear H; intros; elim H; clear H; intros; split with x;
split; auto; intros; generalize (H1 x1 H2); clear H1;
intro; replace ((f x1 + g x1 - (f x0 + g x0)) / (x1 - x0)) with
((f x1 - f x0) * / (x1 - x0) + (g x1 - g x0) * / (x1 - x0)).
assumption. field. elim H2; intros; unfold D_x in H3; elim H3; auto with real.
Qed.
```

$$\cos(0) = 1$$

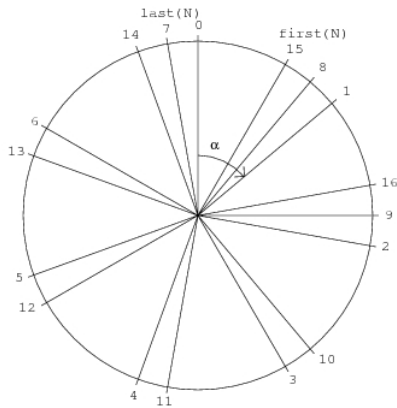
Lemma cos_0 : cos 0 = 1.

Proof.

```
cut (cos_in 0 (cos 0)).
cut (cos_in 0 1).
unfold cos_in in |- *; intros; eapply uniqueness_sum.
apply H0.
apply H.
exact (projT2 exist_cos0).
assert (H := projT2 (exist_cos (Rsqr 0)));
unfold cos in |- *;
  pattern 0 at 1 in |- *; replace 0 with (Rsqr 0);
  [ exact H | apply Rsqr_0 ].
```

Qed.

Le théorème des 3 intervalles



$N=17$, $\alpha=0.139$

Le théorème des 3 intervalles : énoncé

$$\text{after}(N, m) - m =$$

$$\begin{cases} \text{first}(N) & \text{if } 0 \leq m < N - \text{first}(N) \\ \text{first}(N) - \text{last}(N) & \text{if } N - \text{first}(N) \leq m < \text{last}(N) \\ -\text{last}(N) & \text{if } \text{last}(N) \leq m < N \end{cases}$$

Voir démo...

Alors ?

convaincus ?

On peut faire des preuves avec des réels en Coq.

Ne pas hésiter à écrire de tactiques.