8ème Rencontres Arithmétique de l'Informatique Mathématique

Renormalisation d'expansions : une vérification formelle

Sylvie Boldo, Miora Joldes, Jean-Michel Muller, Valentina Popescu

29 juin 2016

8ème Rencontres Arithmétique de l'Informatique Mathématique

Renormalisation d'expansions : une ébauche de vérification formelle

Sylvie Boldo, Miora Joldes, Jean-Michel Muller, Valentina Popescu

29 juin 2016

Disclaimer

Je présente un travail **en cours**.

Les preuves ne sont pas finies !

Quand j'ai prévu cet exposé, il restait quelques brouettes et un lemme technique.

Plan

Motivations

- Expansions flottantes : plus de précision à bas coût.

Motivations

- Expansions flottantes : plus de précision à bas coût.
- Des algorithmes compliqués.

Motivations

- Expansions flottantes : plus de précision à bas coût.
- Des algorithmes compliqués.
- Des preuves très compliquées avec beaucoup de sous-cas.

Motivations

- Expansions flottantes : plus de précision à bas coût.
- Des algorithmes compliqués.
- Des preuves très compliquées avec beaucoup de sous-cas.

$\Rightarrow$ une preuve formelle !

Preuve formelle

La preuve est vérifiée dans ses moindres détails, jusqu'à ce que l'ordinateur l'accepte.

Nous utilisons des assistants de preuves (*formal proof checkers*), c'est-à-dire des programmes qui ne font que vérifier une preuve (et parfois générer une preuve).

En conséquence, le vérificateur est un programme très court (critère de de Bruijn : la correction d'un système entier ne dépend que de la correction d'un très petit « noyau »).

L'assistant de preuves Coq (`http://coq.inria.fr`)

- basé sur l'isomorphisme de Curry-Howard
  (équivalence entre preuve et λ-terme)
  $\Rightarrow$ garantie théorique
- peu d'automatisations
- des bibliothèques fournies, notamment sur $\mathbb{Z}$ et $\mathbb{R}$
- bibliothèque Flocq (B., Melquiond) sur l'arithmétique flottante
- Coq vérifie mécaniquement chaque étape de chaque preuve.
- Le but est d'appliquer successivement des tactiques (application de théorème, réécriture, calcul...) pour modifier ou résoudre un but.
- La preuve est faite en partant de la conclusion.

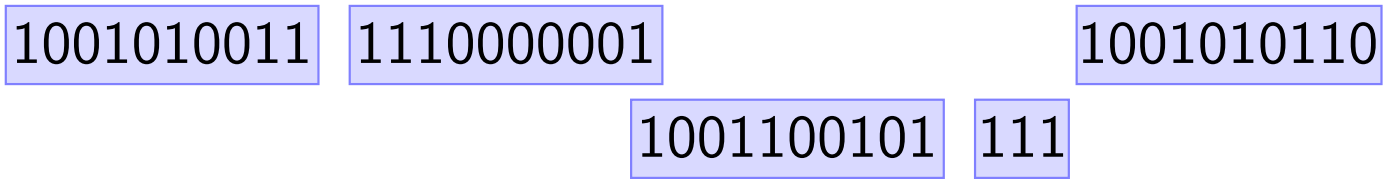
Exemple en Coq 8.4

```
Lemma error_le_plus: forall x y, format x -> format y ->
   (Rabs (round_flt (x+y) - (x+y)) <= bpow (-p)*Rabs (x+y)).
Proof with auto with typeclass_instances.
intros x y Fx Fy.
case (Rle_or_lt (Rabs (x+y)) (bpow (p+emin))); intros H.
rewrite round_generic...
unfold Rminus; rewrite Rplus_opp_r, Rabs_R0; apply Rmult_le_pos;
  [ apply bpow_ge_0 | apply Rabs_pos ].
apply FLT_format_plus_small...
apply Rle_trans with (1:=error_le_half_ulp _ _ _ _).
apply Rle_trans with (/2*(Rabs (x+y) * bpow (1 - p))).
apply Rmult_le_compat_l; [left; apply pos_half_prf | idtac].
apply ulp_FLT_le; rewrite Zplus_comm; now left.
replace (/2) with (bpow (-1)) by reflexivity.
rewrite Rmult_comm, Rmult_assoc, <- bpow_plus.
right; rewrite Rmult_comm; apply f_equal2; apply f_equal; ring.
Qed.
```

Code : mots-clés, énoncé du théorème, tactiques, noms des théorèmes utilisés.

Expansions ?

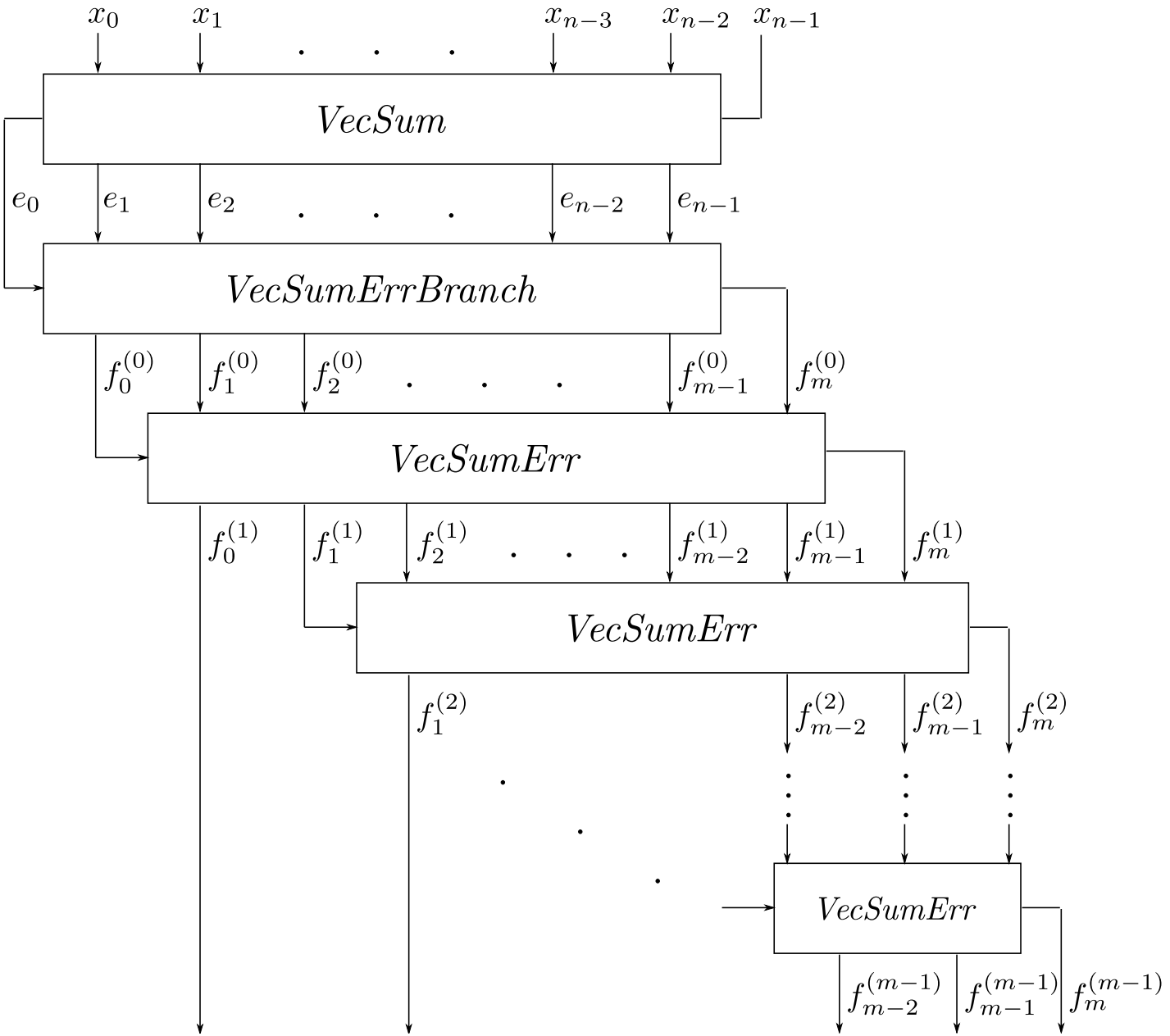
On considère $x = \sum x_i$ avec $x_i \in \mathbb{F}$.



$\hookrightarrow$ opérations ($+, \times, \div, \sqrt{}$, $\ll$ nettoyage $\gg$)
$\hookrightarrow$ preuves (avec possibilité de 0 intermédiaires)

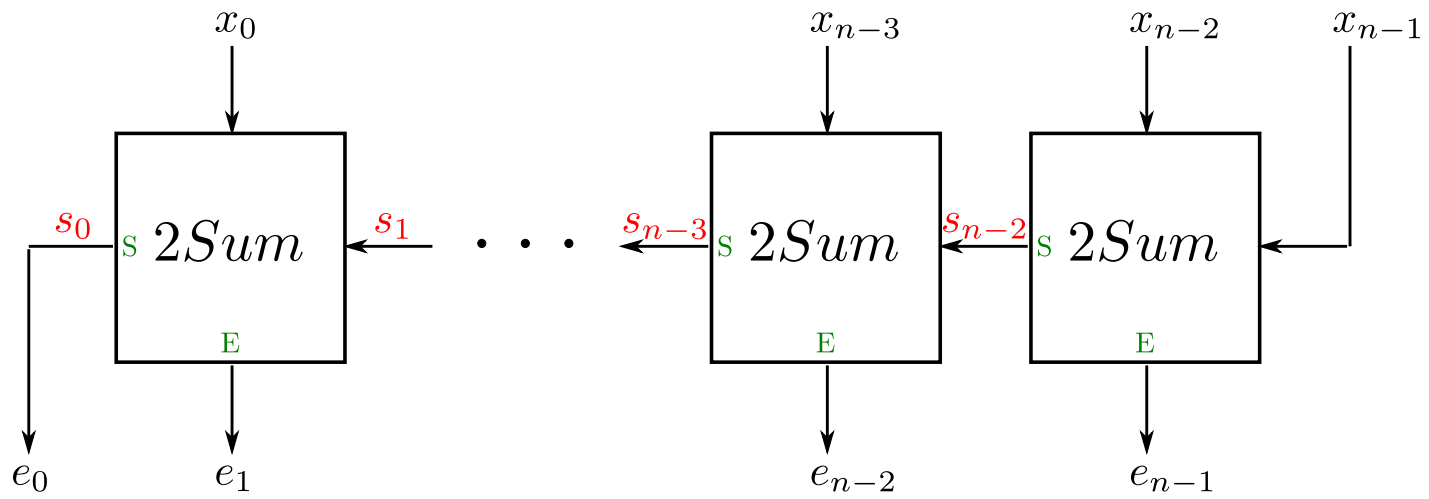
Article **Arithmetic algorithms for extended precision using floating-point expansions** de Mioara Joldes, Olivier Marty, Jean-Michel Muller et Valentina Popescu, IEEE TC, 2016.

Renormalisation



VecSum

Commençons par :



Plan initial

- définir les expansions en Flocq
- comprendre et formaliser les propriétés sur les expansions en entrée et sortie des différents niveaux de l'algorithme
- comprendre et formaliser la preuve

Plan initial

- définir les expansions en Flocq
- comprendre et formaliser les propriétés sur les expansions en entrée et sortie des différents niveaux de l'algorithme
- comprendre et formaliser la preuve

Problèmes attendus :

- dénormalisés ($\not\in$ preuve papier)

Plan initial

- définir les expansions en Flocq
- comprendre et formaliser les propriétés sur les expansions en entrée et sortie des différents niveaux de l'algorithme
- comprendre et formaliser la preuve

Problèmes attendus :

- dénormalisés ($\not\in$ preuve papier)
- zéros intermédiaires

Plan

Base

Choix : une expansion est une **liste** de réels.

$+\!+$ inductions, accès au n-ième élément, liste formée des n premiers éléments. . .

Base

Choix : une expansion est une **liste** de réels.

++ inductions, accès au n-ième élément, liste formée des n premiers éléments...

\- lemmes manquants sur les listes

Base

Choix : une expansion est une **liste** de réels.

++ inductions, accès au n-ième élément, liste formée des n premiers éléments...

- lemmes manquants sur les listes

– problèmes d'itérateurs : `fold_left / right` ne convenaient pas (*out of scope* de cet exposé)

Formalisation des expansions (1/2)

Pour une propriété P donnée entre deux réels ($P : \mathbb{R} \to \mathbb{R} \to \mathit{Prop}$). On définit une expansion comme une liste de flottants tels que deux éléments successifs en enlevant les zéros vérifient P.

Formalisation des expansions (1/2)

Pour une propriété P donnée entre deux réels ($P : \mathbb{R} \to \mathbb{R} \to Prop$). On définit une expansion comme une liste de flottants tels que deux éléments successifs en enlevant les zéros vérifient P.

```
(* Definition of an expansion with a given property *)

Inductive Exp_P (P:R->R->Prop): list R -> Prop :=
  | Exp_Nil : Exp_P P List.nil
  | Exp_One : forall x : R, format x -> Exp_P  P (x :: nil)
  | Exp_Z1 : forall l: list R, Exp_P P l
                  -> Exp_P  P (0 :: l)
  | Exp_Z2 : forall x: R, forall l: list R, Exp_P P (x::l)
                  ->  Exp_P P (x :: ( 0 :: l))
  | Exp_NZ : forall x y: R, forall l: list R, format x
     -> x <> 0 -> y <> 0 -> Exp_P P (y::l)
     -> P x y -> Exp_P P (x :: (y :: l)).
```

Par exemple `Exp_P` $\leq$ définit une liste croissante, en sautant les zéros.

Formalisation des expansions (2/2)

- des tas de lemmes plus ou moins débiles, et

```
Lemma nth_Exp_P: forall (P:R->R->Prop) l, Forall format l ->
(forall i j, (0 <= i < length l)%nat -> (0 <= j < length l)%nat
  -> (i < j)%nat -> nth i l 0 <> 0 -> nth j l 0 <> 0
  -> P  (nth i l 0) (nth j l 0)) -> Exp_P P l.
```

Formalisation des expansions (2/2)

- des tas de lemmes plus ou moins débiles, et

```
Lemma nth_Exp_P: forall (P:R->R->Prop) l, Forall format l ->
(forall i j, (0 <= i < length l)%nat -> (0 <= j < length l)%nat
  -> (i < j)%nat -> nth i l 0 <> 0 -> nth j l 0 <> 0
  -> P  (nth i l 0) (nth j l 0)) -> Exp_P P l.
```

- définition générale pour le VecSum / VecSumErr

Formalisation des expansions (2/2)

- des tas de lemmes plus ou moins débiles, et

```
Lemma nth_Exp_P: forall (P:R->R->Prop) l, Forall format l ->
(forall i j, (0 <= i < length l)%nat -> (0 <= j < length l)%nat
  -> (i < j)%nat -> nth i l 0 <> 0 -> nth j l 0 <> 0
  -> P (nth i l 0) (nth j l 0)) -> Exp_P P l.
```

- définition générale pour le VecSum / VecSumErr
- et les lemmes qui vont avec.

Entrée/sortie du VecSum

En entrée, on a une expansions qui « overlap par au plus d bits » (pour $0 < d \leq p-2$), ce qui devient une expansion avec la propriété :

$$P = \text{ fun } x\ y \Rightarrow |x| < 2^d \mathrm{ulp}(y).$$

Entrée/sortie du VecSum

En entrée, on a une expansions qui « overlap par au plus d bits » (pour $0 < d \leq p-2$), ce qui devient une expansion avec la propriété :

$$P = \text{ fun } x\ y \Rightarrow |x| < 2^d \mathrm{ulp}(y).$$

En sortie, on a une expansion « S-non overlapping (Shewchuck) », ce qui devient une expansion avec la propriété :

$$P = \text{ fun } x\ y \Rightarrow \exists e : Z, \exists n : Z, y = n \times 2^e \wedge |x| < 2^e.$$

Entrée/sortie du VecSum

En entrée, on a une expansions qui « overlap par au plus d bits » (pour $0 < d \leq p-2$), ce qui devient une expansion avec la propriété :

$$P = \text{ fun } x\ y \Rightarrow |x| < 2^d \mathrm{ulp}(y).$$

En sortie, on a une expansion « S-non overlapping (Shewchuck) », ce qui devient une expansion avec la propriété :

$$P = \text{ fun } x\ y \Rightarrow \exists e : Z, \exists n : Z, y = n \times 2^e \wedge |x| < 2^e.$$

```
Definition IVS_P:=(fun x y => Rabs x < bpow d * ulp_flt y).
Definition InputVecSum := Exp_P IVS_P.

Theorem VecSum_correct1: forall l: list R,
   fold_right Rplus 0 l = fold_right Rplus 0 (VecSum l).
intros f; apply VecSum_g_correct1.
Qed.

Theorem VecSum_correct2: forall l: list R,
  InputVecSum l -> OutputVecSum (VecSum l).
Proof.  [.....]
Admitted.
```

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre)

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al.,

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$,

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓

- On peut utiliser des Fast-Two-Sum dans VecSum. ✓

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓

- On peut utiliser des Fast-Two-Sum dans VecSum. ✓
- Et les dénormalisés ? ✗

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓

- On peut utiliser des Fast-Two-Sum dans VecSum. ✓
- Et les dénormalisés ? ✗
  On voit bien les trous dans la preuve formelle !
  J'ai ajouté des `admit` pour l'instant.

Preuves & problèmes

- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

  Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓
- On peut utiliser des Fast-Two-Sum dans VecSum. ✓
- Et les dénormalisés ? ✗
  On voit bien les trous dans la preuve formelle !
  J'ai ajouté des `admit` pour l'instant.
- Et les zéros intermédiaires ?

Preuves & problèmes

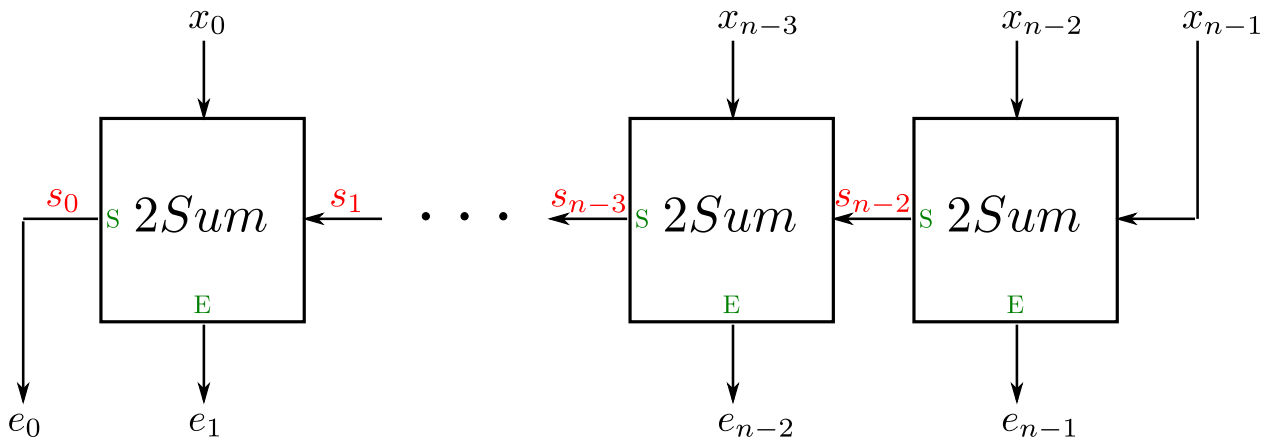
- « D'après Jeannerod et Rump », l'erreur d'une somme flottante est inférieure à ... (cf exposé de Claude-Pierre) ✓
- D'après Joldes et al., ✗

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \cdots \leq 2^d \frac{2^p}{2^p - 1}$$

Je propose plutôt $2^d \frac{2^{p-d}}{2^{p-d}-1}$, mais c'est pas grave ✓

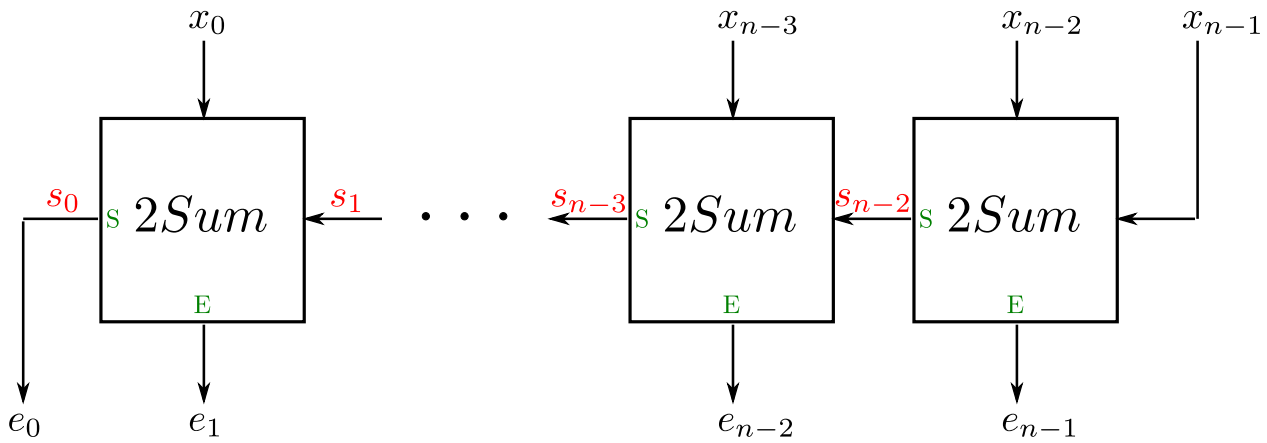
- On peut utiliser des Fast-Two-Sum dans VecSum. ✓
- Et les dénormalisés ? ✗
  On voit bien les trous dans la preuve formelle !
  J'ai ajouté des `admit` pour l'instant.
- Et les zéros intermédiaires ? ✗
  AAAAAAAAAAAAAAAAHHHHHHHHHHHHHHHHHHHHHH

Lemme technique 1/2



Sachant que les x_i sont fortement (dé-)croissants, $|x_{i+1}| < 2^d \mathrm{ulp}(x_i)$ si non nuls, je veux prouver que la suite $\exp(s_i)$ est également croissante, car je m'en sers comme exposant de non-overlapping.

Lemme technique 1/2



Sachant que les x_i sont fortement (dé-)croissants, $|x_{i+1}| < 2^d \mathrm{ulp}(x_i)$ si non nuls, je veux prouver que la suite $\exp(s_i)$ est également croissante, car je m'en sers comme exposant de non-overlapping. ✗

$\hookrightarrow$ non utile dans la preuve initiale

$\hookrightarrow$ après échanges avec JMM, VP, MJ...

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

- $\exp(s_i)$ croissante ✗

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

- $\exp(s_i)$ croissante ✗
- sachant que e_{i-1} et e_j sont non nul, $\exp(s_i) \leq \exp(s_j)$ ✗

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

- $\exp(s_i)$ croissante ✗
- sachant que e_{i-1} et e_j sont non nul, $\exp(s_i) \leq \exp(s_j)$ ✗
- quand $s_i \times x_i \geq 0$ ou $2|s_i| < |x_i|$, alors $\exp(s_i) \leq \exp(s_{i+1})$ ✓

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

- $\exp(s_i)$ croissante ✗
- sachant que e_{i-1} et e_j sont non nul, $\exp(s_i) \leq \exp(s_j)$ ✗
- quand $s_i \times x_i \geq 0$ ou $2|s_i| < |x_i|$, alors $\exp(s_i) \leq \exp(s_{i+1})$ ✓
- quand $s_i \times x_i < 0$ et $|x_i| \leq 2|s_i|$ et $s_{i+1} \neq 0$, alors $\exp(s_i) \leq \exp(s_{i+2})$ ✓

Lemme technique 2/2

On note $s_i = \oplus_{j=0}^{i-1} x_i$ et $e_i = \mathrm{errf}(s_i, x_i)$ (et donc $s_{i+1} + e_i = s_i + x_i$).

- $\exp(s_i)$ croissante ✗
- sachant que e_{i-1} et e_j sont non nul, $\exp(s_i) \leq \exp(s_j)$ ✗
- quand $s_i \times x_i \geq 0$ ou $2|s_i| < |x_i|$, alors $\exp(s_i) \leq \exp(s_{i+1})$ ✓
- quand $s_i \times x_i < 0$ et $|x_i| \leq 2|s_i|$ et $s_{i+1} \neq 0$, alors $\exp(s_i) \leq \exp(s_{i+2})$ ✓
- si on supprime les zéros en entrée, on a

$$\exp(s_i) \leq \max\left(\exp(s_{j-1}), \exp(s_j)\right) \checkmark$$

WIP (perspectives à court terme)

- supprimer les zéros en entrée,
- prouver que $\max(\exp(s_{j-1}), \exp(s_j))$ est un exposant de non-overlapping acceptable,
- trouver des exemples de mes cas bizarres.

Plan

Conclusion 1/2

coqwc me donne :

spec	proof	comments	
310	123	7	AboutFP.v
191	978	205	AboutLists.v
641	405	72	Renormalization.v
1142	1506	284	total

pour un total de 16 définitions, 128 lemmes, 24 admits

Conclusion 2/2

Je m'attendais à

- des lemmes stupides
- des preuves en plus
- des soucis avec les zéros intermédiaires
- trouver des lemmes marrants
  (comme $s_i \neq 0$, sauf si tous les $(x_k)_{k<i}$ sont nuls)

Conclusion 2/2

Je m'attendais à

- des lemmes stupides
- des preuves en plus
- des soucis avec les zéros intermédiaires
- trouver des lemmes marrants
  (comme $s_i \neq 0$, sauf si tous les $(x_k)_{k<i}$ sont nuls)

Je ne m'attendais pas à

- des notations très différentes entre le papier et la preuve formelle
- des inductions, dans des inductions, dans des *case split*
- une erreur dans l'article IEEE-TC
- de GROS soucis avec les zéros intermédiaires

Perspectives + petite annonce

Perspectives :

- finir le lemme technique et vérifier qu'il correspond bien dans la preuve principale
- gérer les dénormalisés
- écrire proprement sur papier ce que j'ai prouvé en Coq

Perspectives + petite annonce

Perspectives :

- finir le lemme technique et vérifier qu'il correspond bien dans la preuve principale
- gérer les dénormalisés
- écrire proprement sur papier ce que j'ai prouvé en Coq

Petite annonce :

Je cherche des gens qui bornent l'erreur d'arrondi sur x par

$$\triangle\left(2^{-53} \times |\circ(x)|\right) \quad \left(\text{voire} + 2^{-1074}\right)$$

en calculant vraiment cette valeur.