Comment dompter un troupeau de flottants sauvages ?

Compilation certifiée des flottants au delà d'IEEE-754

Arthur Correnson - JFLA 2023

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Contexte et motivation (1)

CompCert et la compilation certifiée

Contexte et motivation (1)

CompCert et la compilation certifiée

- La compilation certifiée a pour but de prouver que le sens des programmes est préservé lors de la compilation

Contexte et motivation (1)

CompCert et la compilation certifiée

- La compilation certifiée a pour but de prouver que le sens des programmes est préservé lors de la compilation

  Exemple d'usage: vérifier que le code C embarqué dans un avion est correctement compilé

Contexte et motivation (1)

CompCert et la compilation certifiée

- La compilation certifiée a pour but de prouver que le sens des programmes est préservé lors de la compilation

  Exemple d'usage: vérifier que le code C embarqué dans un avion est correctement compilé

- **CompCert** est un exemple de compilateur certifié pour le langage C

  D'abord développé à l'INRIA, puis, par l'entreprise AbsInt qui commercialise un version industrielle du compilateur

Contexte et motivation (1)

CompCert et la compilation certifiée

$$\mathcal{L}_{\text{source}}$$

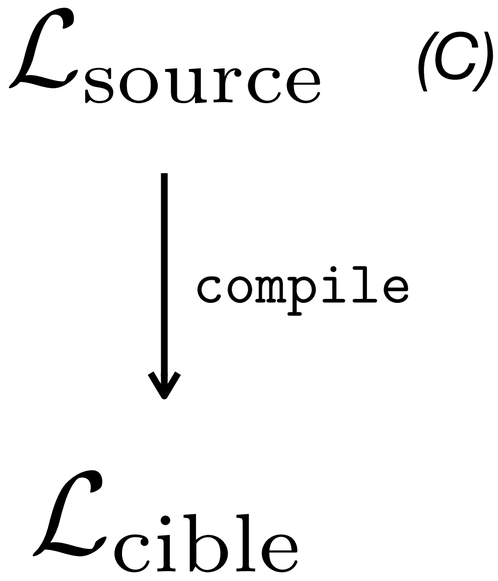
Contexte et motivation (1)

CompCert et la compilation certifiée

$\mathcal{L}_{\text{source}}$ *(C)*

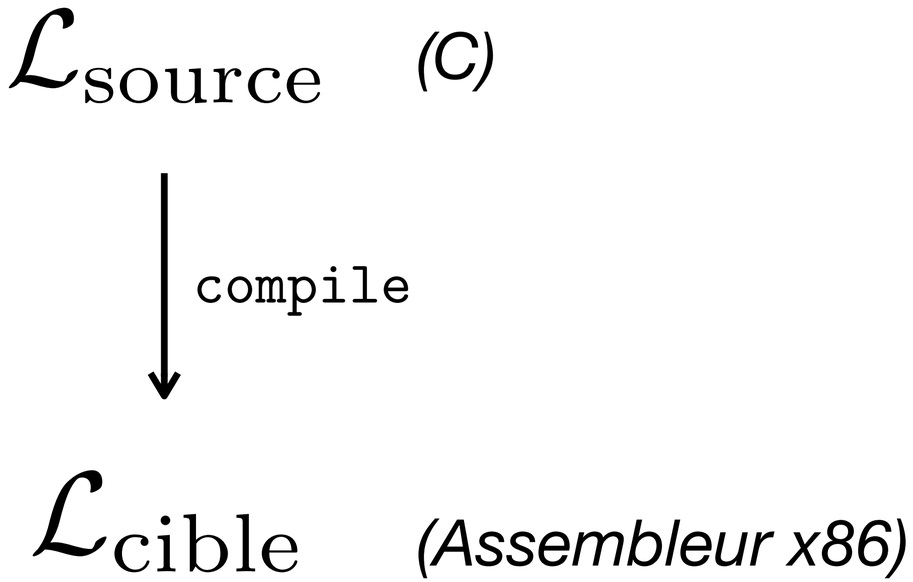
Contexte et motivation (1)

CompCert et la compilation certifiée



Contexte et motivation (1)

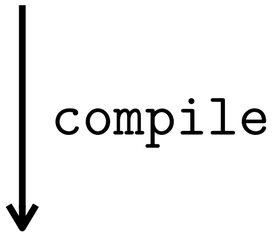
CompCert et la compilation certifiée



Contexte et motivation (1)

CompCert et la compilation certifiée

$\mathcal{L}_{\text{source}}$ *(C)*



$\mathcal{L}_{\text{cible}}$ *(Assembleur x86)*

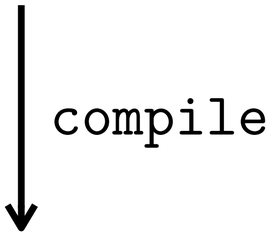
Objectif:

$$\frac{P \in \mathcal{L}_{\text{source}} \qquad P' = \texttt{compile}(P) \in \mathcal{L}_{\text{cible}}}{P \Downarrow_{\text{source}} C \Leftrightarrow P' \Downarrow_{\text{cible}} C}$$

Contexte et motivation (1)

CompCert et la compilation certifiée

$\mathcal{L}_{\text{source}}$ *(C)*



$\mathcal{L}_{\text{cible}}$ *(Assembleur x86)*

Objectif:

$$\frac{P \in \mathcal{L}_{\text{source}} \qquad P' = \texttt{compile}(P) \in \mathcal{L}_{\text{cible}}}{P \Downarrow_{\text{source}} C \Leftrightarrow P' \Downarrow_{\text{cible}} C}$$

Prouvé en Coq !

Contexte et motivation (2)

Compilation certifiée des nombres flottants

- Que disent les sémantiques en matière d'arithmétique flottante ?

Contexte et motivation (2)

Compilation certifiée des nombres flottants

- Que disent les sémantiques en matière d'arithmétique flottante ?

$$\texttt{printf("\%f", 0.1 + 0.2 + 0.3)} \Downarrow \boxed{?}$$

Contexte et motivation (2)

Compilation certifiée des nombres flottants

- Que disent les sémantiques en matière d'arithmétique flottante ?

$$\texttt{printf("\%f", 0.1 + 0.2 + 0.3)} \Downarrow \boxed{?}$$

- Particulièrement important dans les domaines avec des forts besoins en calcul numérique

Les flottants dans les langages de programmation

Les flottants dans les langages de programmation

Ce que dit le langage C

> *The accuracy of the floating-point operations (`+`, `-`, `*`, `/`) and of the library functions in `<math.h>` and `<complex.h>` that return floating-point results is implementation-defined. The implementation may state that the accuracy is unknown*

Extrait de ISO/IEC 9899, section 5.2.4.2.2 "characteristics of floating types"

Les flottants dans les langages de programmation

Ce que dit le langage C

« Pour les flottants, faites un peu comme vous voulez ! »

> *The accuracy of the floating-point operatio[illegible] and of the library functions in* `<math.h>` *and* `<complex.h>` *that return floating-point results is implementation-defined. The implementation may state that the accuracy is unknown*

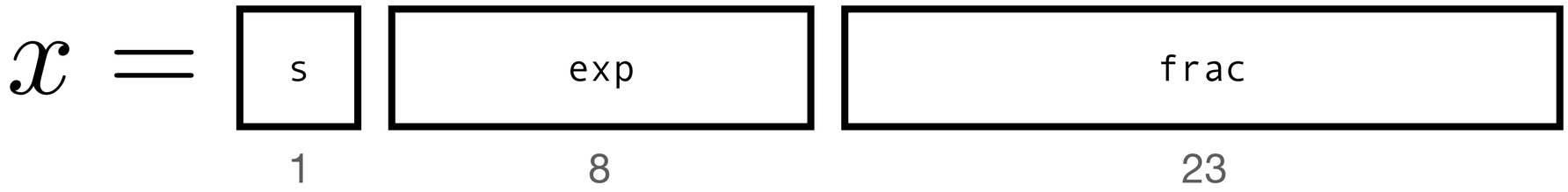
Extrait de ISO/IEC 9899, section 5.2.4.2.2 "characteristics of floating types"

Un standard pour les flottants

Ce que dit la norme IEEE-754

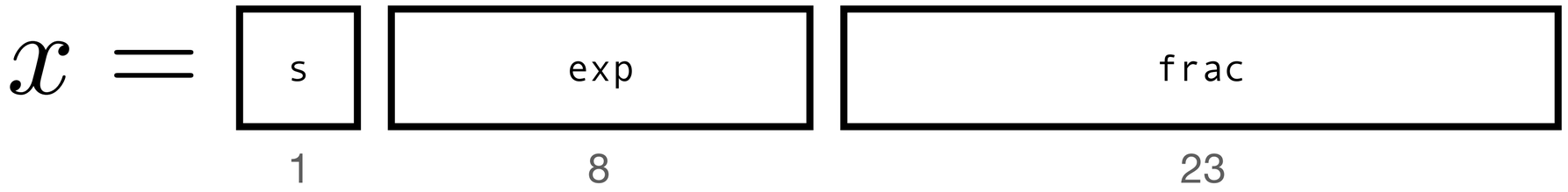
Un standard pour les flottants

Ce que dit la norme IEEE-754



Un standard pour les flottants

Ce que dit la norme IEEE-754



Valeurs	Interprétation	condition
NaN	valeur indéterminée	$\texttt{exp} = 255$, $\texttt{frac} > 0$
$+\infty, -\infty$	infinis signés en fonction du bit `s`	$\texttt{exp} = 255$, $\texttt{frac} = 0$
$+0, -0$	zéros signés en fonction du bit `s`	$\texttt{exp} = 0$, $\texttt{frac} = 0$
Nombres normalisés	$(-1)^{\texttt{s}} \times 2^{\texttt{exp}-127} \times (1.\texttt{frac})_2$	$1 \leq \texttt{exp} < 255$
Nombres dénormalisés	$(-1)^{\texttt{s}} \times 2^{\texttt{exp}-127} \times (0.\texttt{frac})_2$	$\texttt{exp} = 0$, $\texttt{frac} > 0$

Un standard pour les flottants

Ce que dit la norme IEEE-754

$$\texttt{round}_m : \mathbb{R} \to \texttt{float}$$

Un standard pour les flottants

Ce que dit la norme IEEE-754

$$\texttt{round}_m : \mathbb{R} \to \texttt{float}$$

mode d'arrondi *(Au plus proche, vers 0, vers -inf, vers +inf, …)*

Un standard pour les flottants

Ce que dit la norme IEEE-754

$$\texttt{round}_m : \mathbb{R} \to \texttt{float}$$

$$x +_{\texttt{float}} y = \texttt{round}_m(x +_{\mathbb{R}} y)$$

Un standard pour les flottants

Ce que dit la norme IEEE-754

$$\texttt{round}_m : \mathbb{R} \to \texttt{float}$$

« Les opérations se comportent comme l'arrondi de leur pendant réel »

$$x +_{\texttt{float}} y = \texttt{round}_m(x +_{\mathbb{R}} y)$$

Un standard pour les flottants

Ce que dit la norme IEEE-754

Avantages:

Un standard pour les flottants

Ce que dit la norme IEEE-754

Avantages:

On a une sémantique claire

ET un *manuel de référence* pour son implémentation

Un standard pour les flottants

Ce que dit la norme IEEE-754

Avantages:

On a une sémantique claire

ET un *manuel de référence* pour son implémentation

Inconvénients:

Différents modes d'arrondi et, donc, **différentes sémantiques**

Un standard pour les flottants

Ce que dit la norme IEEE-754

Avantages:

On a une sémantique claire

ET un *manuel de référence* pour son implémentation

Inconvénients:

Différents modes d'arrondi et, donc, **différentes sémantiques**

```
printf("%f", 0.1 + 0.2 + 0.3)
```
⇓ ?

Un standard pour les flottants

Ce que dit la norme IEEE-754

Avantages:

On a une sémantique claire

ET un *manuel de référence* pour son implémentation

Inconvénients:

Différents modes d'arrondi et, donc, **différentes sémantiques**

`printf("%f", 0.1 + 0.2 + 0.3)` ⇓ ?

Avec quel mode d'arrondi ?

Les flottants en *hardware*

Les flottants en *hardware*

Le cas des unités de calcul flottantes *SPE*

> *Embedded floating-point operations do not produce +Inf, –Inf, NaN, or a denormalized number. [. . .] the interrupt handler is responsible for delivering IEEE 754–compliant behavior if desired*

Extrait de Signal Processing Engine (SPE) Programming Environments Manual,
section 3.3.1.4 about IEEE Std 754 Compliance

Les flottants en *hardware*

Le cas des unités de calcul flottantes *SPE*

> *Embedded floating-point operations do not produce +Inf, –Inf, NaN, or a denormalized number. [...] the interrupt handler is responsible for delivering IEEE 754–compliant behavior if desired*

Extrait de Signal Processing Engine (SPE) P
section 3.3.1.4 about IEEE

« Faites presque comme IEEE-754, mais traitez tous les cas particuliers d'une manière complètement différente »

Les flottants en *hardware*

Le cas des unités de calcul flottantes *SPE*

Avantages:

- Meilleurs coûts de fabrication
- Meilleures performances (parfois)

Les flottants en *hardware*

Le cas des unités de calcul flottantes *SPE*

Avantages:

- Meilleurs coûts de fabrication
- Meilleures performances (parfois)

Inconvénients:

- Des sémantiques différentes selon l’environnement d’exécution matériel

Quelles conclusions tirer ?

Une arithmétique intrinsèquement difficile…

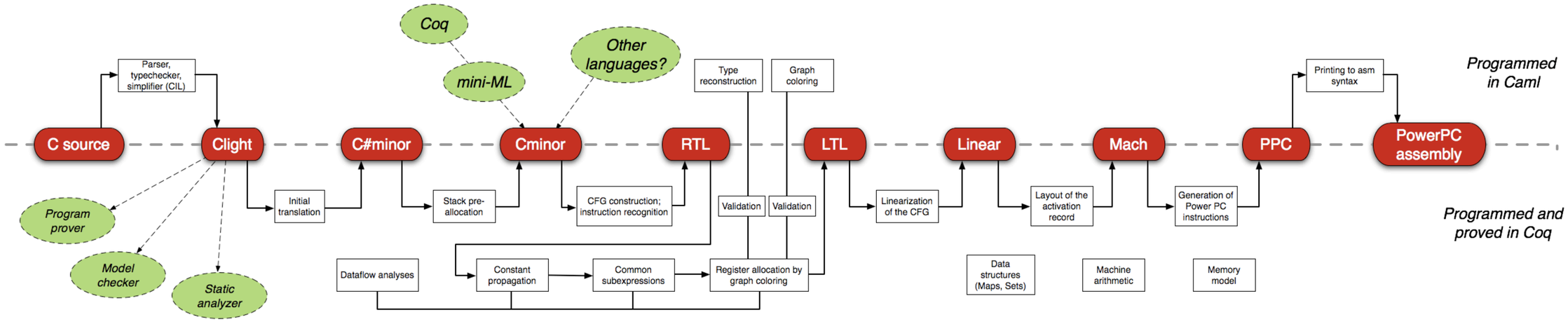
… *relativement* standardisée (IEEE-754) …

… mais dont l’interprétation diffère d’un environnement à l’autre

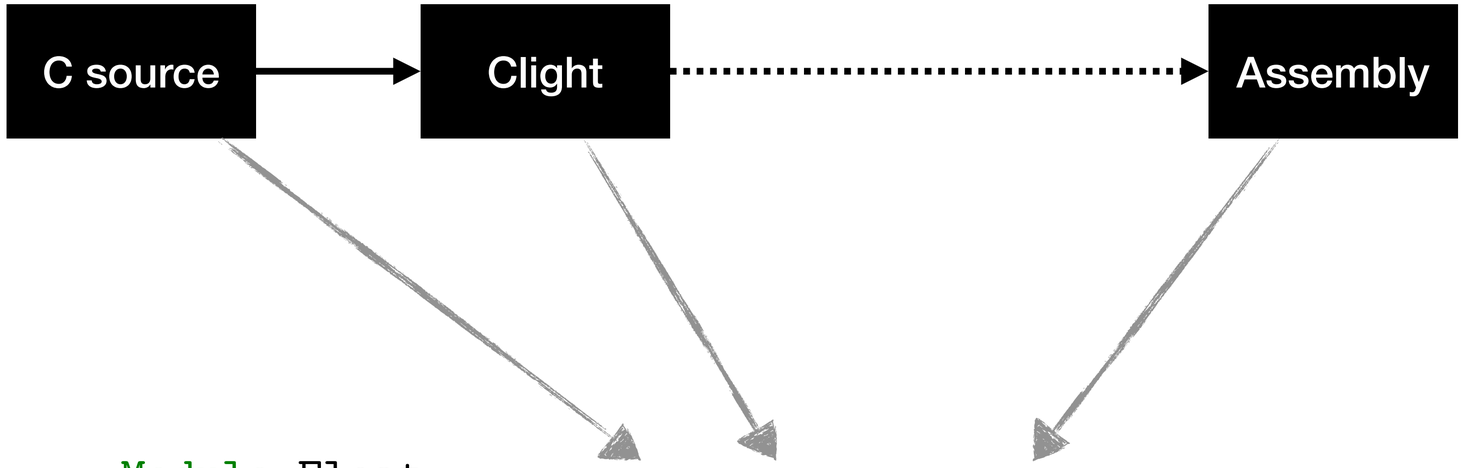
Comment formaliser le comportement des flottants dans les programmes ?

L’approche CompCert

L’approche CompCert

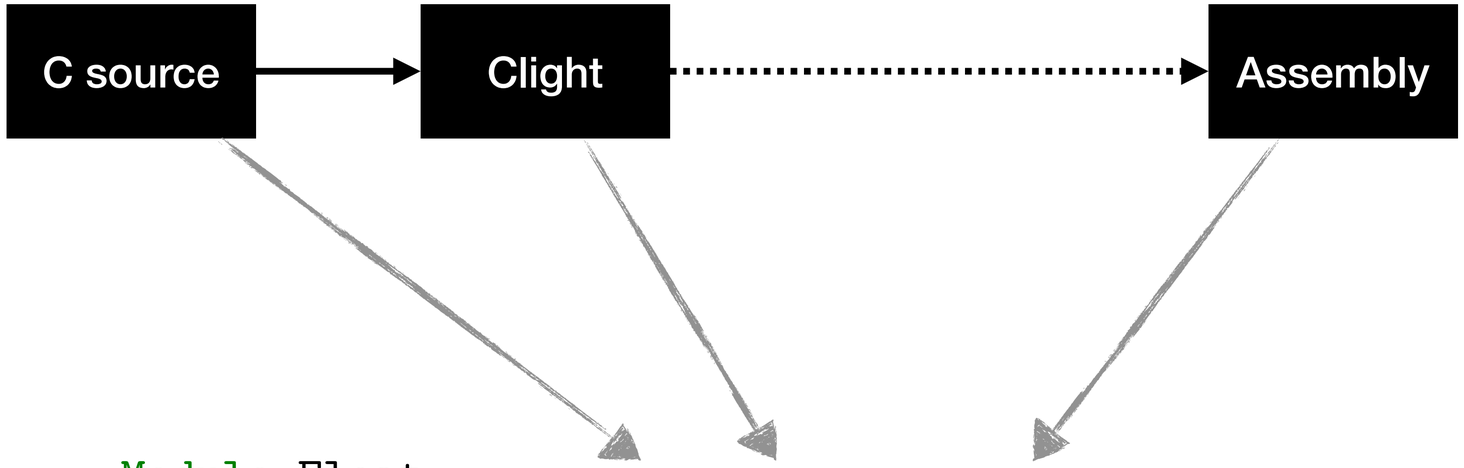


L’approche CompCert




```
Module Float.
  Definition float : Type := ...
  Definition add (x y : float) : float := ...
  Definition mul (x y : float) : float := ...
  ...
  Theorem add_comm : forall (x y : float), add x y = add y x.
End Float.
```

L’approche CompCert




```
Module Float.
  Definition float : Type := ?
  Definition add (x y : float) : float := ?
  Definition mul (x y : float) : float := ?
  ...
  Theorem add_comm : forall (x y : float), add x y = add y x.
End Float.
```

Méthode 1: Axiomatiser les flottants

Méthode 1: Axiomatiser les flottants

```
Module Float.
  Parameter float : Type.
```

Méthode 1: Axiomatiser les flottants

```
Module Float.
  Parameter float : Type.
  Parameter add : float -> float -> float.
  Parameter mul : float -> float -> float.
```

Méthode 1: Axiomatiser les flottants

```
Module Float.
  Parameter float : Type.
  Parameter add : float -> float -> float.
  Parameter mul : float -> float -> float.
  ...
  Axiom add_comm : forall (x y : float), add x y = add x y.
  ...
End Float.
```

Méthode 1: Axiomatiser les flottants

Avantages:

Facile à mettre en oeuvre

Pas besoin de modéliser tous les modèles de calcul différents

Méthode 1: Axiomatiser les flottants

Avantages:

- Facile à mettre en oeuvre
- Pas besoin de modéliser tous les modèles de calcul différents

Inconvénients:

- Il faut des axiomes *raisonnables*
- Pas de calcul possible

Méthode 2: Imposer un standard

- On choisi d'abord un modèle de calcul flottant

  Par exemple: IEEE-754 avec arrondi au plus proche

- On en développe une implémentation logicielle

  Par exemple: la bibliothèque Flocq

- On **prouve** les résultats d'arithmétique souhaités

Méthode 2: Imposer un standard

Avantages:

On gagne la possibilité de faire du calcul !

Méthode 2: Imposer un standard

Avantages:

On gagne la possibilité de faire du calcul !

Inconvénients:

On perd en généralité

Vers plus de généricité

Vers plus de généricité

Idéalement on souhaiterait:

Plus de souplesse, y compris quand on fixe la norme IEEE

Vers plus de généricité

Idéalement on souhaiterait:

Plus de souplesse, y compris quand on fixe la norme IEEE

Les défis:

1. Ne pas casser toute la chaine de compilation
2. Identifier les paramètres que l'on peut faire varier
3. Formaliser *sans erreur* les différents modèles de calcul

Méthode 3: *Fonctoriser* les sémantiques ?

Example: IEEE + tous les modes d'arrondi

```
Module Type MODE.
  Parameter round_mode : mode.
End MODE.

Module Float(M : MODE).
...
End Float.
```

Méthode 3: *Fonctoriser* les sémantiques ?

Example: IEEE + tous les modes d'arrondi

```
Module Type MODE.
  Parameter round_mode : mode.
End MODE.

Module Float(M : MODE).
...
End Float.
```

Méthode 3: *Fonctoriser* les sémantiques ?

Question:

Est-ce que tous théorèmes utilisés dans la preuve de correction sont indépendants du mode d'arrondi ?

Méthode 3: *Fonctoriser* les sémantiques ?

Question:

Est-ce que tous théorèmes utilisés dans la preuve de correction sont indépendants du mode d'arrondi ?

```
Module Float(M : MODE).
  ...
  Theorem add_comm: forall (x y : float), add x y = add y x.
  Proof.
    ...
  Qed.
  ...
End Float.
```

La preuve fait maintenant référence à un mode d'arrondi arbitraire !

Méthode 3: *Fonctoriser* les sémantiques ?

Question:

Est-ce que tous théorèmes utilisés dans la preuve de correction sont indépendants du mode d'arrondi ?

Réponse:

Contre toute attente, **oui** ! (Ou presque)

Méthode 3: *Fonctoriser* les sémantiques ?

Question:

Est-ce que tous théorèmes utilisés dans la preuve de correction sont indépendants du mode d'arrondi ?

Réponse:

Contre toute attente, **oui** ! (Ou presque)

Exception:

Certains résultats utiles pour les conversions d'entiers vers flottants ne sont pas vrais en mode « arrondi vers zéro »

Méthode 3: *Fonctoriser* les sémantiques ?

```
Module Type MODE.
  Parameter round_mode : mode.
  Axiom valid_round_mode:
    round_mode <> round_down.
End MODE.
```

Méthode 3: *Fonctoriser* les sémantiques ?

En pratique:

On réalise plusieurs instances du foncteur

Méthode 3: *Fonctoriser* les sémantiques ?

En pratique:

On réalise plusieurs instances du foncteur

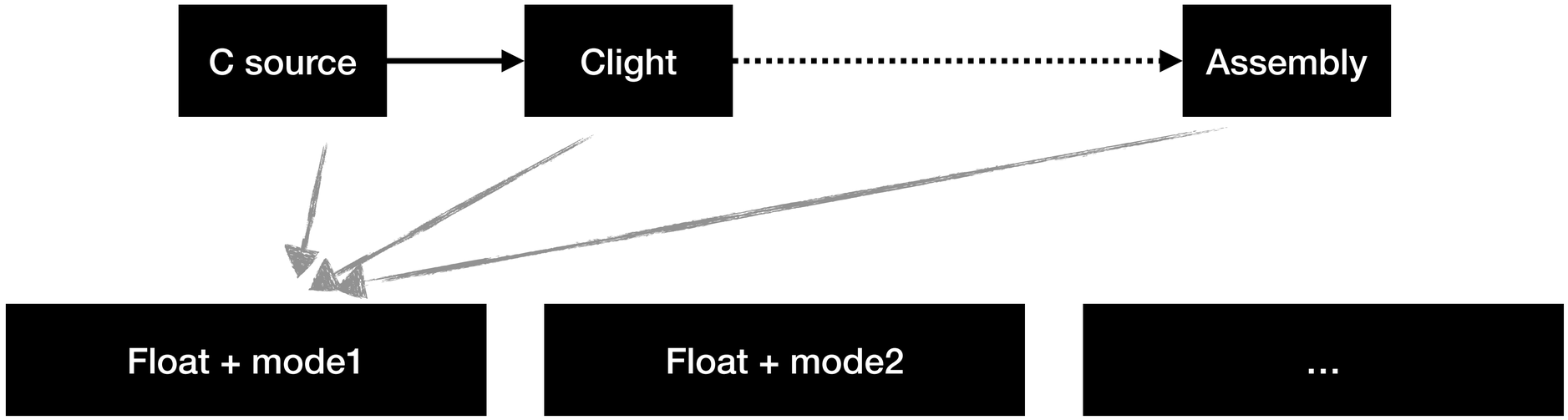
Au moment de la compilation des sources de CompCert, on choisit l'instance de son choix

Méthode 3: *Fonctoriser* les sémantiques ?

En pratique:

On réalise plusieurs instances du foncteur

Au moment de la compilation des sources de CompCert, on choisit l'instance de son choix

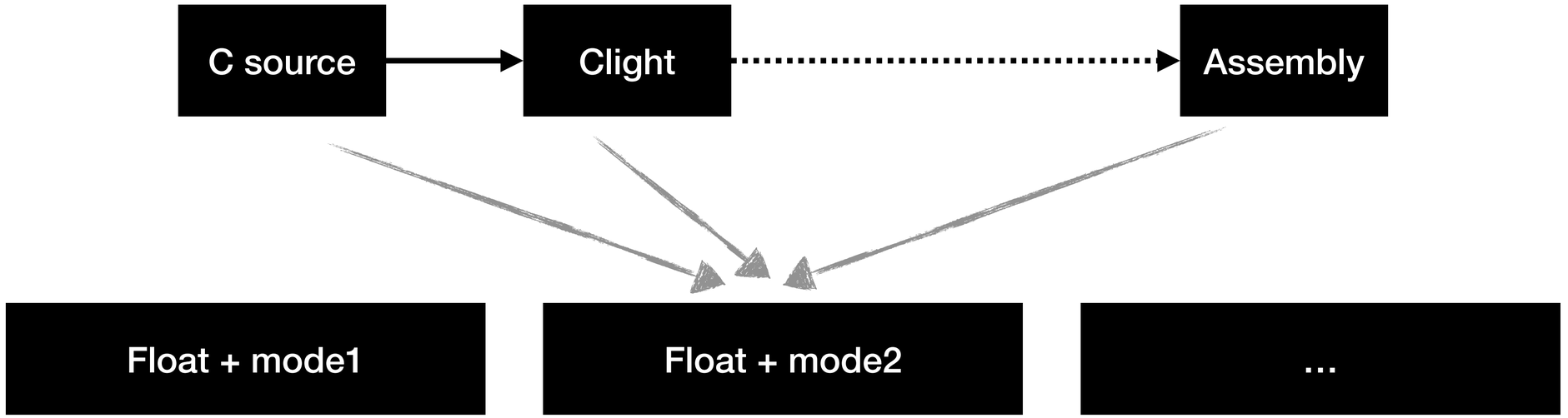


Méthode 3: *Fonctoriser* les sémantiques ?

En pratique:

On réalise plusieurs instances du foncteur

Au moment de la compilation des sources de CompCert, on choisit l'instance de son choix



Méthode 3: *Fonctoriser* les sémantiques ?

Limites:

Impossible d’exprimer des modèles de calcul **trop différents**

Nécessité de de réaliser plusieurs instances

Méthode 3: *Fonctoriser* les sémantiques ?

Limites:

Impossible d'exprimer des modèles de calcul **trop différents**

Nécessité de de réaliser plusieurs instances

Tentative ?

Rendre le module qui implémentent les flottants complètement interchangeable ?

Formaliser différent modèles de calcul

Comment modéliser d’autres modèles de calcul flottants

Si possible en gardant l’aspect **executable**

ET tout en restant fidèle aux documents normatifs

Les défis de la formalization des standards

Les défis de la formalization des standards

> *Each of the computational operations that return a numeric result specified by this standard shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then rounded that intermediate result, if necessary, to fit in the destination's format (see 4 and 7). Clause 6 augments the following specifications to cover* ± 0, $\pm\infty$, *and NaN.*

Les défis de la formalization des standards

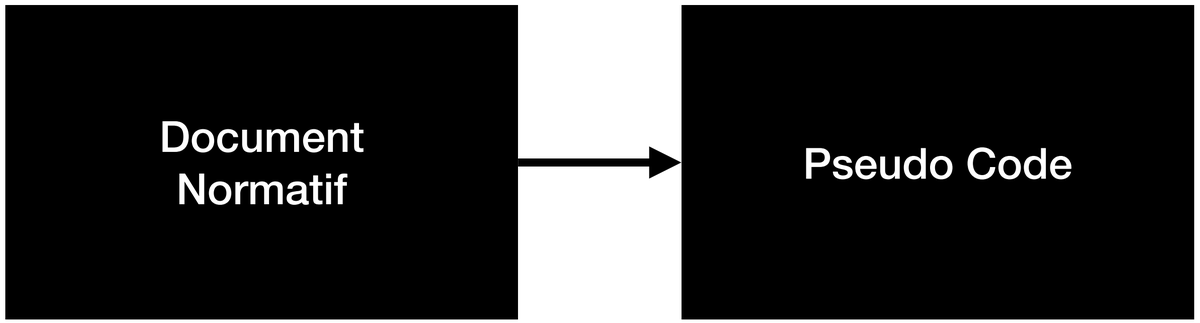
Each of the computational operations that return a numeric result specified by this standard shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then rounded that intermediate result, if necessary, to fit in the destination's format (see 4 and 7). Clause 6 augments the following specifications to cover ± 0, $\pm\infty$, *and NaN.*

Coq ?

Une approche expérimentale

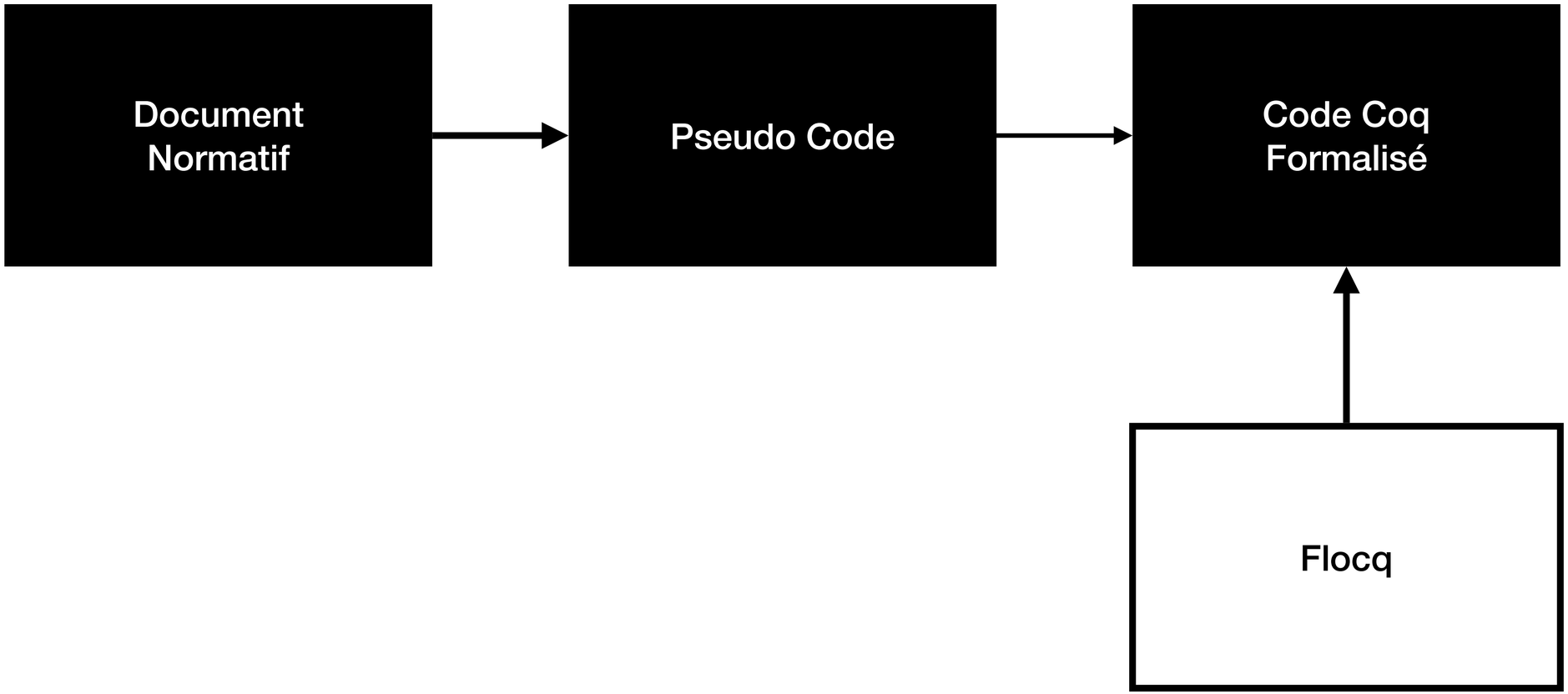
Document
Normatif

Une approche expérimentale

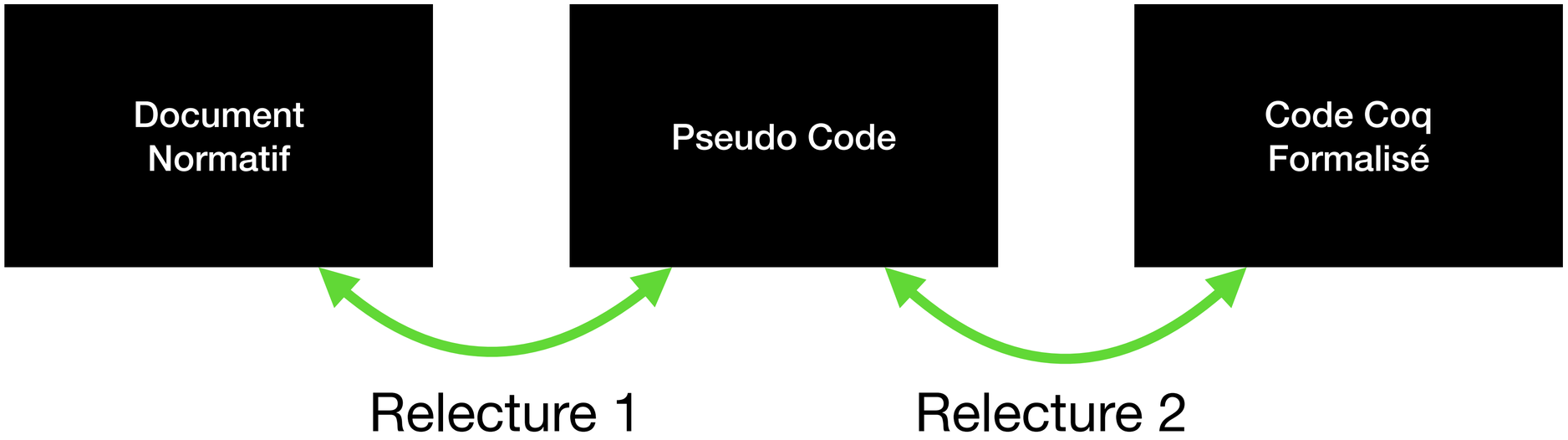


Une approche expérimentale

Une approche expérimentale



Une approche expérimentale



Conclusion

- Un véritable défi de formalisation de la sémantique des programmes flottants
- Plusieurs approches existantes
  - Des approches souples par **axiomatisation** qui *omettent* les singularités des flottants
  - Des approches plus restrictives en **fixant un modèle** de calcul
- Nouvelles approches expérimentales
  - Modèles de calcul paramétrables
  - Sémantiques paramétrables par un modèle de calcul