



# Programmation en nombres entiers et compilation certifiée

Journée Cedric  
30 janvier 2007

Sandrine Blazy

Eric Soutif



## Cadre du projet

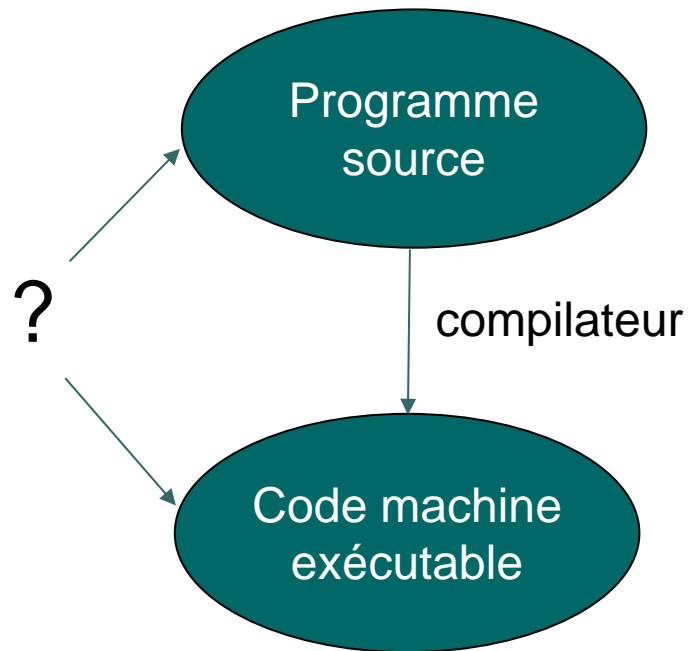
- Projet ANR Compcert
- Vérification formelle de compilateurs optimisants pour logiciel embarqué critique
- Co-encadrement d'un mémoire de Master



# La compilation

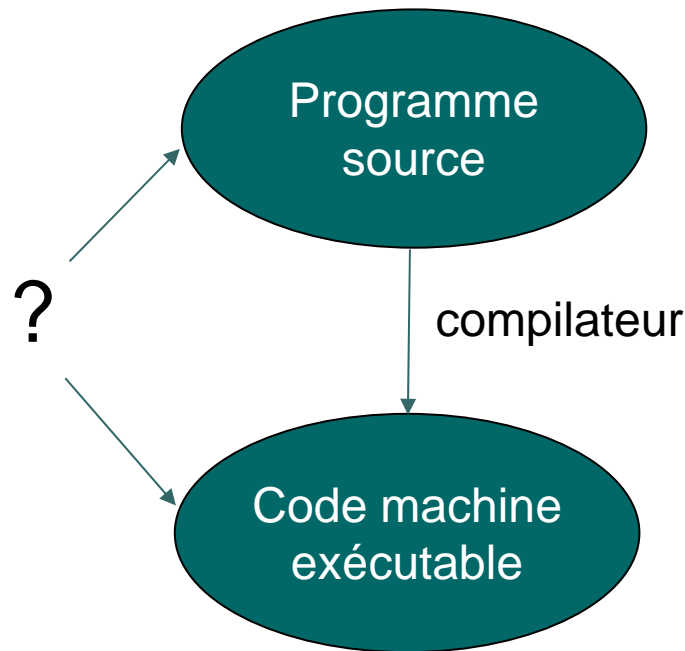
- Au sens large : toute traduction automatique d'un langage informatique vers un autre.
- Au sens restreint : traduction **efficace** (“optimisante”) d'un langage **source** (compréhensible par les programmeurs) vers un langage **machine** (compréhensible par le *hardware*).
- Un domaine mature :
  - Bientôt 50 ans ! (Fortran I : 1957)
  - Énorme corpus d'algorithmes (optimisations).
  - Nombreux compilateurs qui effectuent des transformations très subtiles.

# Faites-vous confiance à votre compilateur ?



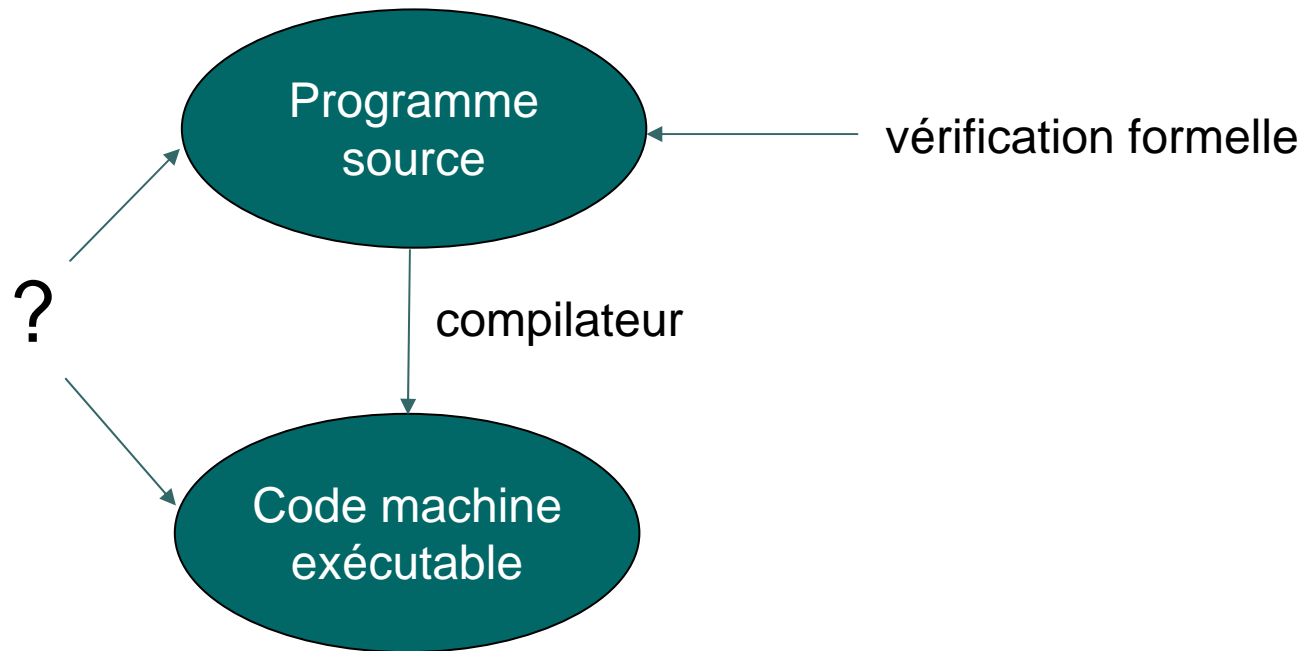
- Un bug dans le compilateur peut faire produire du code machine faux à partir d'un programme correct.

# Faites-vous confiance à votre compilateur ?

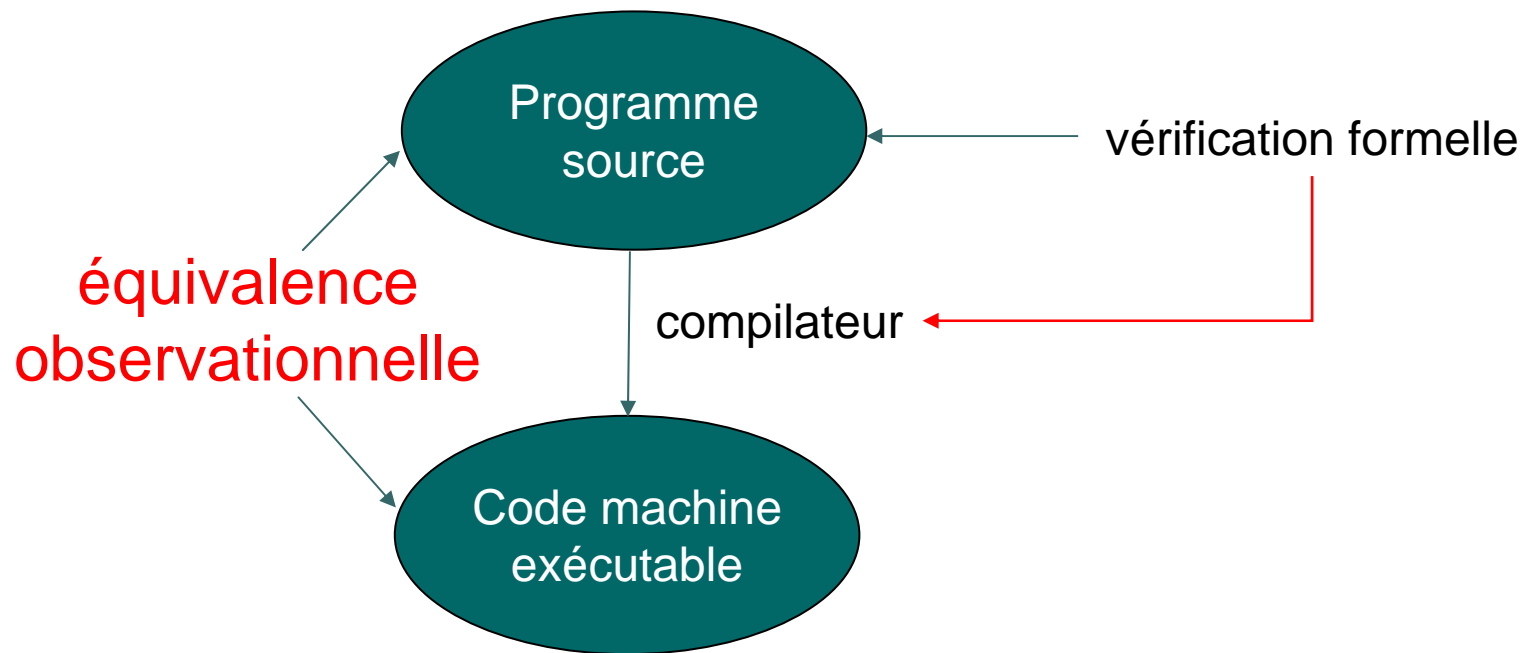


- **Compilateur formellement vérifié :**
- Garantit que le code produit se comporte comme prescrit par la sémantique du programme source.

# Faites-vous confiance à votre compilateur ?



# Faites-vous confiance à votre compilateur ?





# La vérification formelle de compilateurs

Appliquer les méthodes formelles au compilateur lui-même pour établir un théorème de **préservation sémantique** :

## Théorème

*Pour tous les codes source  $S$ ,  
si le compilateur transforme  $S$  en le code machine  $C$ ,  
sans signaler d'erreur de compilation,  
et si  $S$  a une sémantique bien définie,  
alors  $C$  a la même sémantique que  $S$  à équivalence observationnelle près.*

Prouvé sur machine à l'aide de l'assistant à la preuve Coq.



# Allocation de registres

## Algorithme d'Appel et George

- Le meilleur algorithme d'allocation de registres.
- L'algorithme le plus complexe du compilateur certifié.
- N'a pas été prouvé directement, mais validé a posteriori (spécification et preuve trop difficiles à faire sur machine).



# Allocation de registres

## Algorithme d'Appel et George (suite)

- Il consiste en :
  - Une analyse de la portée des variables (*à tel point du programme, quelles sont les variables vivantes ?*) : **liveness analysis**.
  - Une analyse des mouvements entre la mémoire et les registres du processeur (*à tel point du programme, telle variable vivante doit-elle être en mémoire ou dans un registre ?*) : **spilling analysis – approche par la PLNE**
  - Une analyse des mouvements de registre à registre : **coalescing analysis** – approche heuristique

# Allocation de registres

Un exemple tiré de « Correctness of ILP-based Register Allocation » (Naik et Palsberg, 2005)

```
0 : a := 0
1 : a := a + 1
2 : b := b - c
3 : if b < c goto 4 else 1
4 : return a
```

(a) programme source

```
0 : a := 0
1 : load r, a
   r := r + 1
2 : store a, r
   load r, b
   r := r - c
3 : store b, r
   if b < c goto 4 else 1
4 : return a
```

(b) programme cible  
sous-optimal

```
0 : r := 0
1 : r := r + 1
2 : store a, r
   load r, b
   r := r - c
3 : if r < c goto 4
   else {
       store b, r
       load r, a
       goto 1
   }
4 : return a
```

(c) programme cible  
optimal



# Allocation de registres

## Utilisation de la PLNE

- Considérons une variable  $v$  vivante au point de programme  $p$  (point entre deux instructions). La variable  $v$  peut :
  - Arriver à  $p$  dans un registre et sortir de  $p$  dans un registre :  $r_{p,v}$
  - Arriver à  $p$  en mémoire et y rester :  $m_{p,v}$
  - Arriver en registre et sortir en mémoire (stored) :  $s_{p,v}$
  - Arriver en mémoire et sortir en registre (loaded) :  $l_{p,v}$



# Allocation de registres

## Utilisation de la PLNE

- On identifie alors un ensemble de contraintes linéaires à respecter :

$$l_{p,v} + r_{p,v} + s_{p,v} + m_{p,v} = 1$$

$$\sum_{v \text{ vivante à } p} r_{p,v} + l_{p,v} \leq K$$

$$\sum_{v \text{ vivante à } p} r_{p,v} + s_{p,v} \leq K$$

⋮

- Fonction objectif : minimisation de l'estimation du temps d'exécution du programme, tenant compte du coût des mouvements mémoire-registre et des opérations du processeur



## Idée du projet

- Proposer un autre algorithme utilisant la PLNE, et tout aussi efficace.
- Le valider a posteriori ou le spécifier formellement ?