Assembleur Certifié

Roman Delgado

Université Pierre et Marie Curie

31/05/2017

Introduction

Objectifs du projet

- Apprentissage du langage Coq
- Réaliser un assembleur certifié

Pourquoi certifier un assembleur

- Bugs KVM (décodage des instructions)
- Fin de la chaîne de compilation

Programmes certifiés

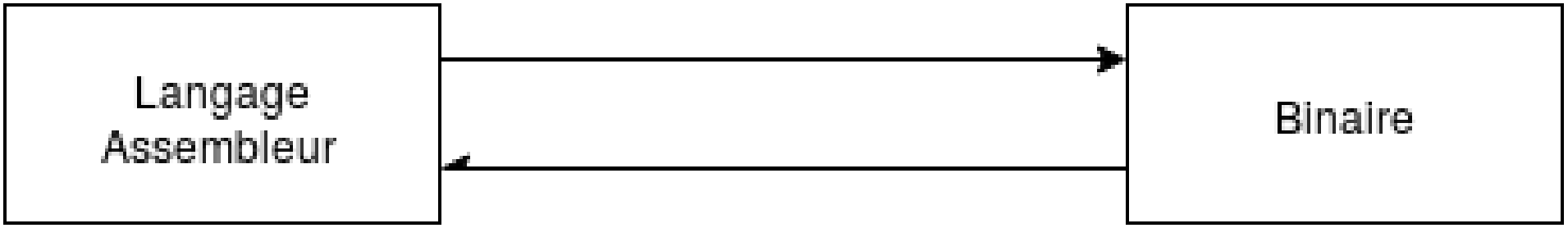
Qu'est-ce qu'un programme certifié ?

- Une spécification formelle
- Une preuve que le programme respecte sa spécification

Programmation certifiée dans Coq

- Coq est un assistant de preuve
- Programme écrit avec Coq
- Preuves également vérifiées par Coq
- Programme exécuté = programme prouvé

Certifier un Assembleur



Fonctions Inverses

- Decodage(Encodage(y)) = y
- Encodage(Decodage(x)) = x $\qquad \forall x \in \mathit{Def}(\mathit{Decodage})$

x86 vs MMIX

x86

- Taille des instructions variable
- Une même instruction possède différents encodages
- De nombreux champs dans une instruction (tag, ect.)

MMIX

- Jeu d'instructions RISC (taille des instructions fixée)
- Format des instructions très simple

Partialité

Monade option

- Mécanisme pour gérer les fonctions partielles
- ```
  Inductive option (A : Type) : Type :=
    Some : A → option A
  | None : option A.
  ```

Exemple fonction de décodage

```
Definition decode (bi : binary_instruction) : option instruction :=
```

Impact sur les théorèmes

```
Lemma decode_encode : forall (bi : binary_instruction)
                             (i : instruction),
    decode bi = Some i → encode i = Some bi.
```
```

Récursion

Exemple Ocaml

```
let rec n_bit k =
  match k with
  | 0 → []
  | n → let l = n_bit (k / 2) in
        (n mod 2) :: l
```

Exemple Coq

```
Fixpoint n_bit (n : nat) (k : nat) : option (list bool) :=
    match n with
      | 0 => match k with
             | 0 => Some []
             | S _ => None
             end
      | S n' => match n_bit n' (Nat.div2 k) with
                  | None => None
                  | Some l => Some (Nat.odd k :: l)
                end
    end.
```

Réflexion

Théorèmes

- ```
  Theorem lookup_lookdown : forall (n : nat) (t : tag) ,
      lookup t encdec = Some n → lookdown n encdec = Some t.
  ```

```
Theorem lookdown_lookup : forall (n : nat) (t : tag),
    lookdown n encdec = Some t → lookup t encdec = Some n.
```

Avantages

- Terme de preuve plus petit
- Plus d'études de cas pour chaque nouveau théorème
```

Conclusion

Réalisations

- Conversion $\mathbb{N}$ vers liste $\mathbb{B}$
- Fonctions d'encodage et de décodage
- Encodage de flux d'instructions et décodage d'une suite de bits

Vers un assembleur x86

- Représentation des données
- Rendre le programme actuel plus compositionnel