Compilateur C vers clauses de Horn

Paul RAYNAUD M1 Informatique
Laboratoire : VERIMAG
Equipe : PACSS
Enseignants référents :
Michël PÉRIN; David MONNIAUX
Soutenance TER (Magistère)

Pourquoi vérifier des programmes ?

- Omniprésence de l'informatique

  - Les ordinateurs sont présent en masse dans notre société et nous aide dans la vie courante, cependant les bogues ne sont pas rares.

- Dans des systèmes critiques

  - Santé, l'aéronautique, énergie, information, finance

  (- Il est vital que ces secteurs soient à l'abris de pannes et de mauvais fonctionnements. Par conséquent il est necessaire de les prouver)

  Transition :

  La technique de preuve de programme est connue . Depuis 1969 (Hoare & Floyd & Dijkstra) la recherche porte sur l'automatisation.

Outils existant

- SeaHorn un outil réalisant la vérification de programme
  - Basé sur le compilateur LLVM, sémantique ambigüe, traduction logique non documentée
- Choix d'un compilateur CompCert C
  - Une sémantique bien définie (en coq)
  - Pas de miscompilations
  - Base solide pour prouver la traduction en formule logique

    (travail à long terme)

Objectif : Du code C à des implications logiques

- Pourquoi ?
  - Générer automatiquement les triplets de Hoare à partir du code source
  - Utiliser un Solveur Modulo Théorie (SMT) pour décider si les implications sont valides.

    Oral :

    - On peut générer automatiquement les triplet de Hoare …
    - aujoud'hui les SMT sont efficaces sur de nombreuses théorie (algèbre linéaire, polynômes, intervalles, vecteurs de bits, logique booléenne, logique de séparation pour le « traitement » des pointeurs)
- Comment ?
  - Choix d'un compilateur/sémantique
  - Besoin de construire/ récupérer un Graphe de Flôt de Contrôle (CFG)

Objectif

```
int main (void) {
        int x=3;
        if( x == 6){}
        else{x = 6;}
        return 0;
}
```

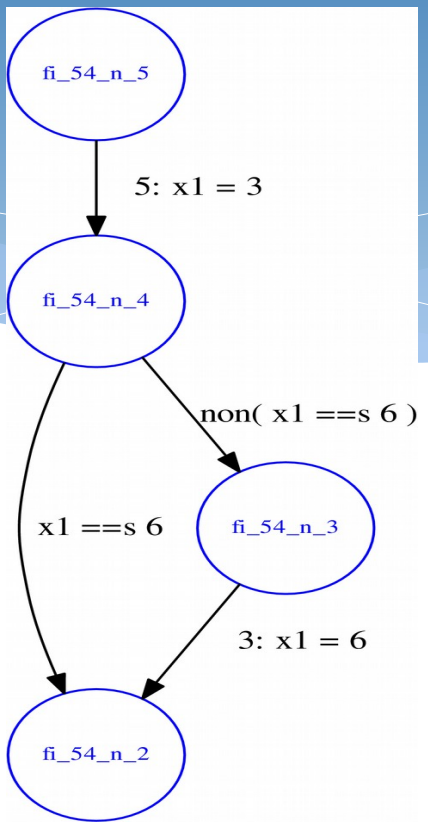
Formule logique générée:

$\rho(x1) = x1 \in \mathbb{N}$,
$\gamma(x1) = (x1 = 6)$,
$\rho(x1) \Rightarrow (((3 = 6) \Rightarrow \gamma(3)) \wedge ((3 \neq 6) \Rightarrow \gamma(6)))$

SMT

Travail réalisé lors du TER

```
int main (void) {
        int x=3;
        if( x == 6){}
        else{x = 6;}
        return 0;
}
```



Travail réalisé:

S'intégrer dans un compilateur industriel parmis 13 représentation intermédiaire

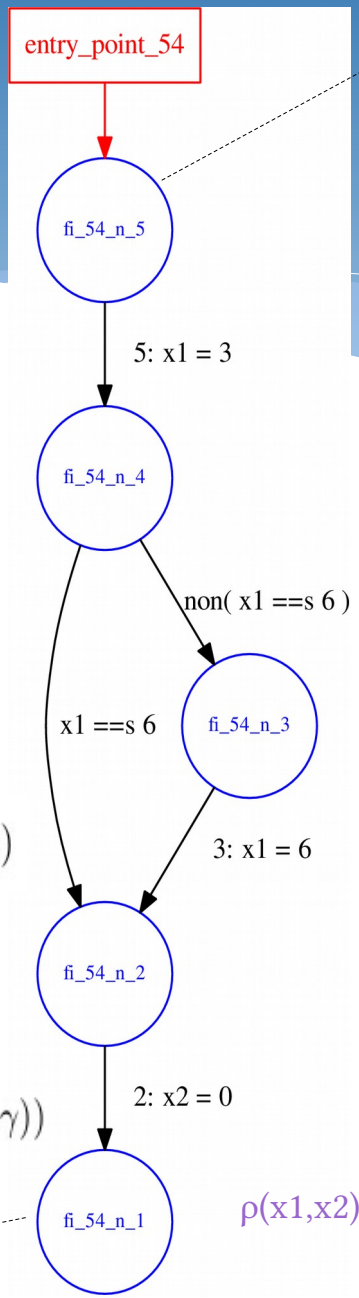
Comprendre la représentation RTL de CompCert

Travail du magistère : Génération des implications logiques

- Rappel de la logique de Hoare :

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$ $\forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma)$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\ if\ C\ then\ S1\ else\ S2,\ \gamma) =$ $(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

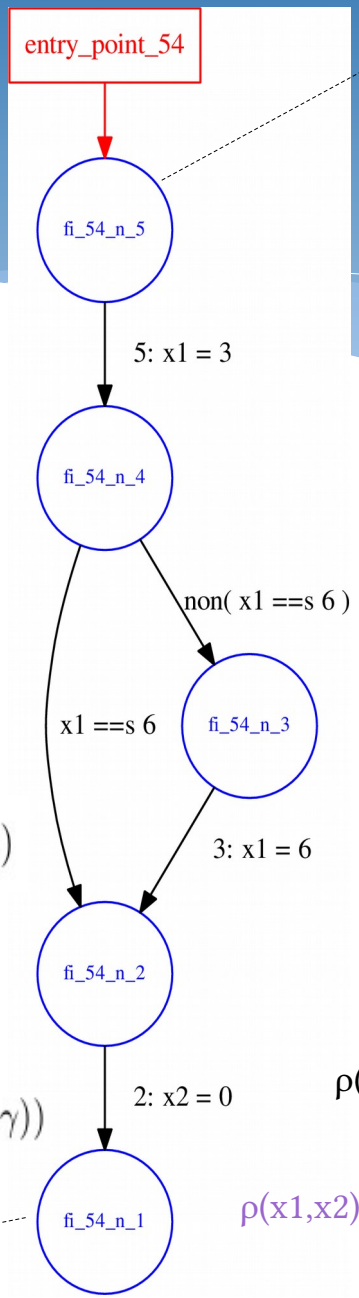
LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$
$\forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma)$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\ if\ C\ then\ S1\ else\ S2,\ \gamma) =$
$(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

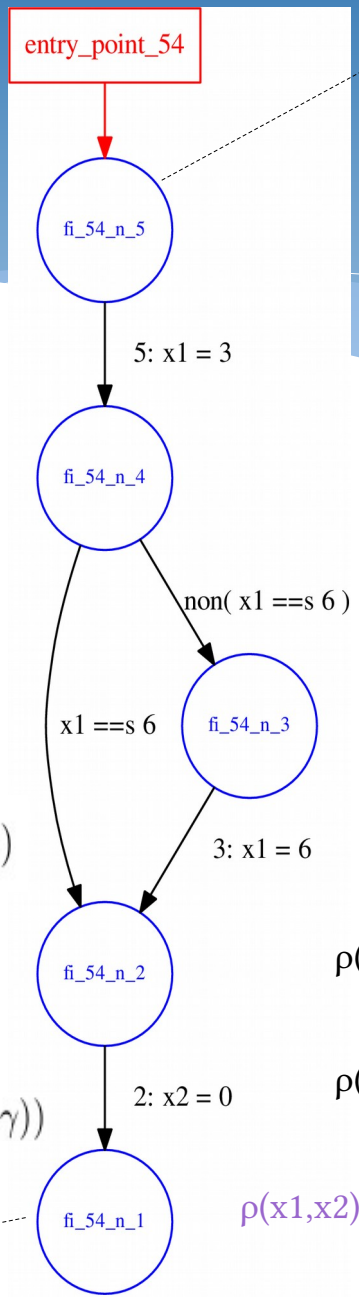
LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$
$\forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma)$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\ if\ C\ then\ S1\ else\ S2,\ \gamma) =$
$(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 , γ(x1, 0))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

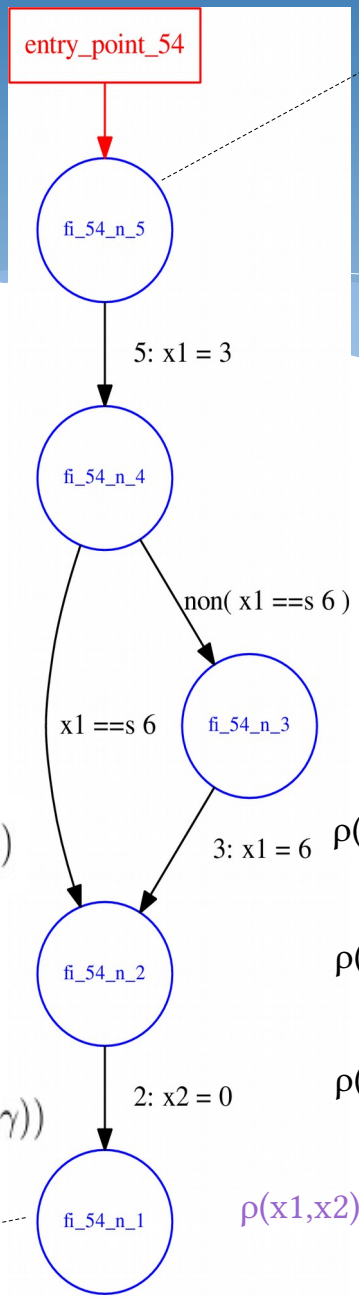
LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$ $\quad \forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma))$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\,if\ C\ then\ S1\ else\ S2,\ \gamma) =$ $(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 , WP(if (x1 = 6) then else x1 := 6 , γ(x1, 0)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 , γ(x1, 0))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

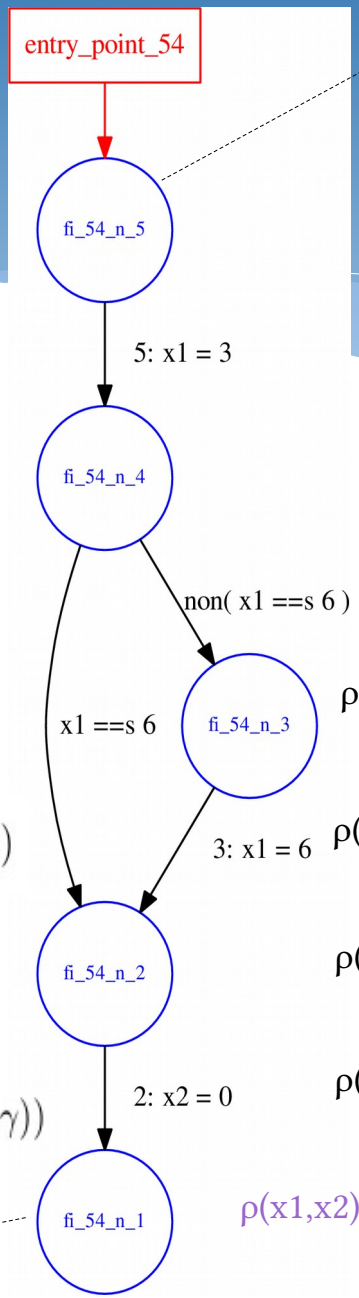
LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$
$\forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma))$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\,if\ C\ then\ S1\ else\ S2,\ \gamma) =$
$(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 , x=6 ⟹ γ(x1, 0) ∧ x1≠6 ⟹WP(x1 :=6, γ(x1, 0)))

(3)

ρ(x1,x2) ⟹WP (x1 :=3 , WP(if (x1 = 6) then else x1 := 6 , γ(x1, 0)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 , γ(x1, 0))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

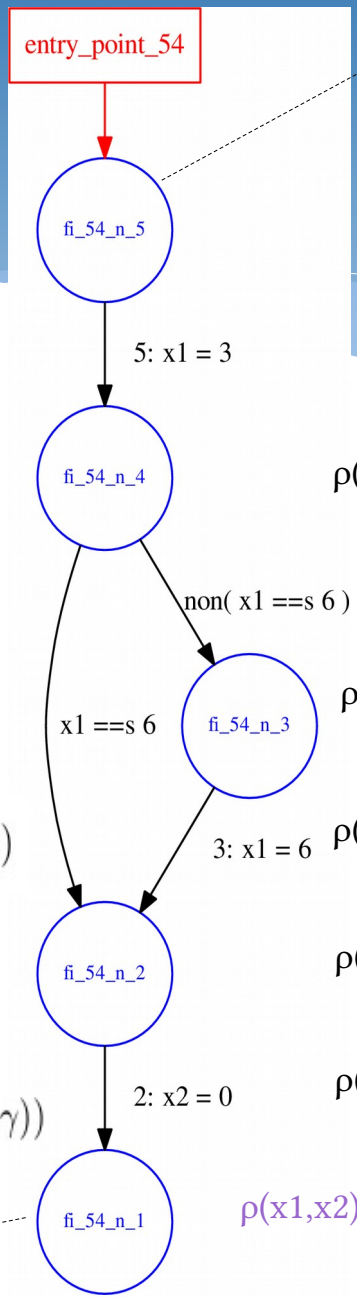
LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple

```
main() {
   5:  x1 = 3
   4:  if (x1 ==s 6) goto 2 else goto 3
   3:  x1 = 6
   2:  x2 = 0
   1:  return x2
```



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$ $\quad \forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma))$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\,if\ C\ then\ S1\ else\ S2,\ \gamma) =$ $(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹WP (x1 :=3 , x1 = 6 ⟹ γ(x1, 0) ∧ x1 ≠ 6 ⟹ γ(x1, 6))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 , x=6 ⟹ γ(x1, 0) ∧ x1≠6 ⟹WP(x1 :=6, γ(x1, 0)))

(3)

ρ(x1,x2) ⟹WP (x1 :=3 , WP(if (x1 = 6) then else x1 := 6 , γ(x1, 0)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 , γ(x1, 0))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

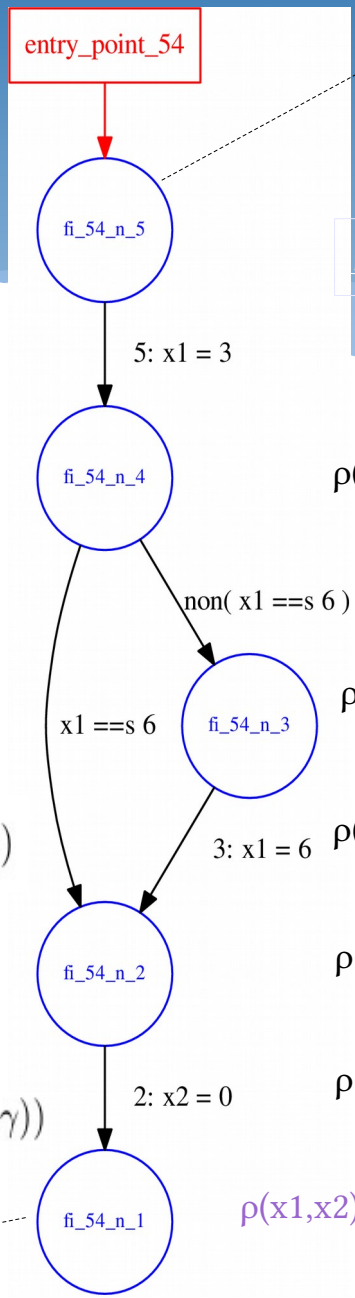
ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Exemple



assume(ρ(x1,x2), ρ(x1,x2) = x1 ∈ ℕ

Assert (γ), γ(x1, x2) = (x1 = 6)

Pas complètement sur que ce soit une clause de horn

Rappel des règles:

(0): $\{\rho\}programme\{\gamma\}$ $\quad \forall x, \rho \implies WP(programme, \gamma)$

(1): $WP(S1; S2, \gamma) = WP(S1, WP(S2, \gamma))$

(2): $WP(x := e, \gamma) = \gamma[x \leftarrow e]$

(3): $WP(\,if\ C\ then\ S1\ else\ S2,\ \gamma) =$ $(C \implies WP(S1, \gamma)) \wedge (\neg C \implies WP(S2, \gamma))$

ρ(x1,x2) ⟹ (((3 = 6) ⟹ γ(3, 0)) ∧ ((3 ≠ 6)⟹ γ(6, 0)))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 , x1 = 6 ⟹ γ(x1, 0) ∧ x1 ≠ 6 ⟹ γ(x1, 6))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 , x=6 ⟹ γ(x1, 0) ∧ x1≠6 ⟹WP(x1 :=6, γ(x1, 0)))

(3)

ρ(x1,x2) ⟹WP (x1 :=3 , WP(if (x1 = 6) then else x1 := 6 , γ(x1, 0)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 , γ(x1, 0))

(2)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6,WP(x2 := 0, γ(x1, x2)))

(1)

ρ(x1,x2) ⟹WP (x1 :=3 ; if (x1 = 6) then else x1 := 6 ; x2 := 0, ψ(x1, x2))

LA BASE

(0)

{ρ} progamme {γ} ⟶ ∀x1,x2 , ρ(x1,x2) ⟹WP (progamme,γ)

Explication du travail fourni

```
int main (void) {
        int x=3;
        if( x == 6){}
        else{x = 6;}
        return 0;
```

Compilé par compcert

```
main() {
    5:  x1 = 3
    4:  if (x1 ==s 6) goto 2 else goto 3
    3:  x1 = 6
    2:  x2 = 0
    1:  return x2
```

En brouillon, tentative de montrer les résultats

```
(AST.Gfun
   (AST.Internal
      { RTL.fn_sig =
        { AST.sig_args = []; sig_res = (Some AST.Tint);
          sig_cc =
          { AST.cc_vararg = false; cc_unproto = false;
            cc_structret = false }
          };
        fn_params = []; fn_stacksize = BinNums.Z0;
        fn_code =
        (Maps.PTree.Node (
           (Maps.PTree.Node (
              (Maps.PTree.Node (Maps.PTree.Leaf,
                 (Some (RTL.Icond (
                          (Op.Ccompimm (Integers.Ceq,
                             (BinNums.Zpos
                                (BinNums.Coq_x0
                                   (BinNums.Coq_xI BinNums.Coq_xH)))
                             )),
                          [BinNums.Coq_xH],
                          (BinNums.Coq_x0 BinNums.Coq_xH),
                          (BinNums.Coq_xI BinNums.Coq_xH)))),
                 Maps.PTree.Leaf)),
              (Some (RTL.Iop ((Op.Ointconst BinNums.Z0), [],
                       (BinNums.Coq_x0 BinNums.Coq_xH), BinNums.Coq_xH))),
              Maps.PTree.Leaf)),
           (Some (RTL.Ireturn (Some (BinNums.Coq_x0 BinNums.Coq_xH)))),
           (Maps.PTree.Node (
              (Maps.PTree.Node (Maps.PTree.Leaf,
                 (Some (RTL.Iop (
                          (Op.Ointconst
                             (BinNums.Zpos
                                (BinNums.Coq_xI BinNums.Coq_xH))),
                          [], BinNums.Coq_xH,
                          (BinNums.Coq_x0
                             (BinNums.Coq_x0 BinNums.Coq_xH))
                          ))),
                 Maps.PTree.Leaf)),
              (Some (RTL.Iop (
                       (Op.Ointconst
                          (BinNums.Zpos
                             (BinNums.Coq_x0
                                (BinNums.Coq_xI BinNums.Coq_xH)))),
                       [], BinNums.Coq_xH,
                       (BinNums.Coq_x0 BinNums.Coq_xH)))),
              Maps.PTree.Leaf))
```

Travail restant à fournir dans le futur

- Réaliser un première version simplifié

  - L'objectif de mon stage : ...

  ou

  - Réaliser la génération des clauses de Horn pour un code source, en supportant un sous-ensemble des instructions du C. (ce qui est le sujet de mon stage)

- Etendre la première version

  - Réaliser toutes les instructions supportées par Compcert

  ou

  - Un outil pouvant supporter toutes les instructions définies dans Compcert.

- Vers une intégration dans Compcert
  - Prouver l'outil en coq et l'intégrer a Compcert

Questions ?