Détermination de propriétés de flots de données pour l'amélioration du temps d'exécution pire-cas

Candidat
Jordy Ruiz

Encadrant
Hugues Cassé

20 juin 2014

Sommaire

Introduction

Équipe TRACES

Estimation du pire temps d'exécution (WCET)

- But : surestimer le temps d'exécution d'une partie de programme
- Exemples
  - Le frein de voiture s'activera au pire 50ms après la commande
  - L'algorithme prendra une décision en moins d'1s
  - ...

Estimation du pire temps d'exécution (WCET)

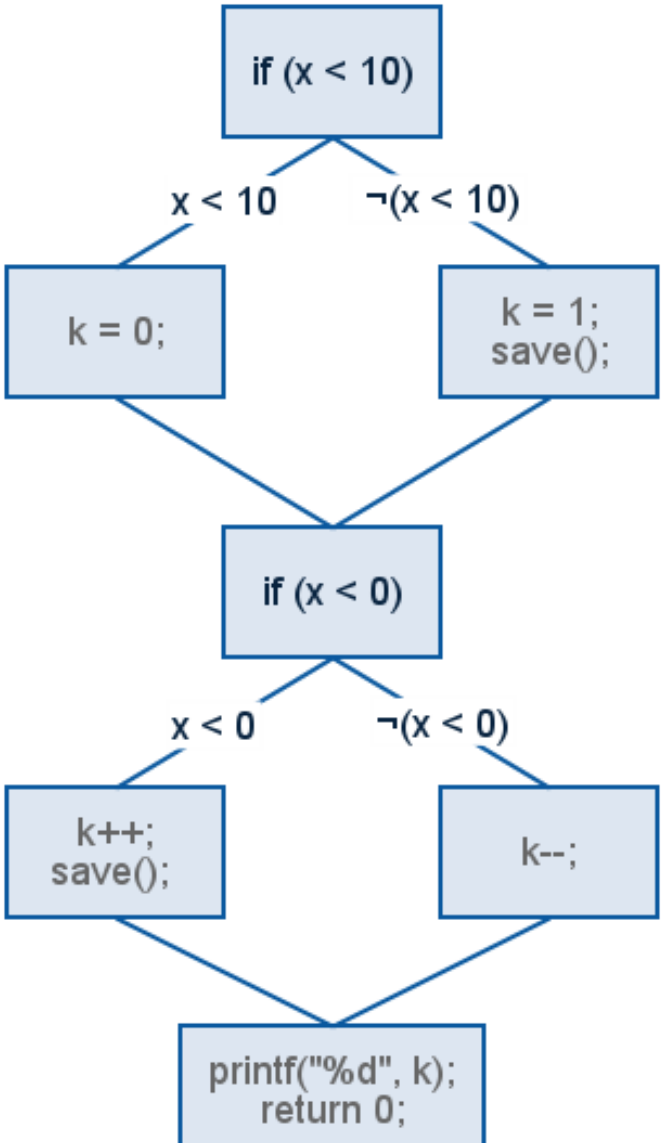
- But : surestimer le temps d'exécution d'une partie de programme
- Exemples
  - Le frein de voiture s'activera au pire 50ms après la commande
  - L'algorithme prendra une décision en moins d'1s
  - ...
- Systèmes temps-réel critiques : les mesures ne suffisent pas, il faut une **preuve** !

Estimation du pire temps d'exécution (WCET)

- But : surestimer le temps d'exécution d'une partie de programme
- Exemples
  - Le frein de voiture s'activera au pire 50ms après la commande
  - L'algorithme prendra une décision en moins d'1s
  - ...
- Systèmes temps-réel critiques : les mesures ne suffisent pas, il faut une **preuve** !
- Calcul du pire-temps : maximisation en ILP
  - $WCET = \max \sum x_i \ t_i$
    $+$ contraintes matérielles $+$ contraintes logicielles

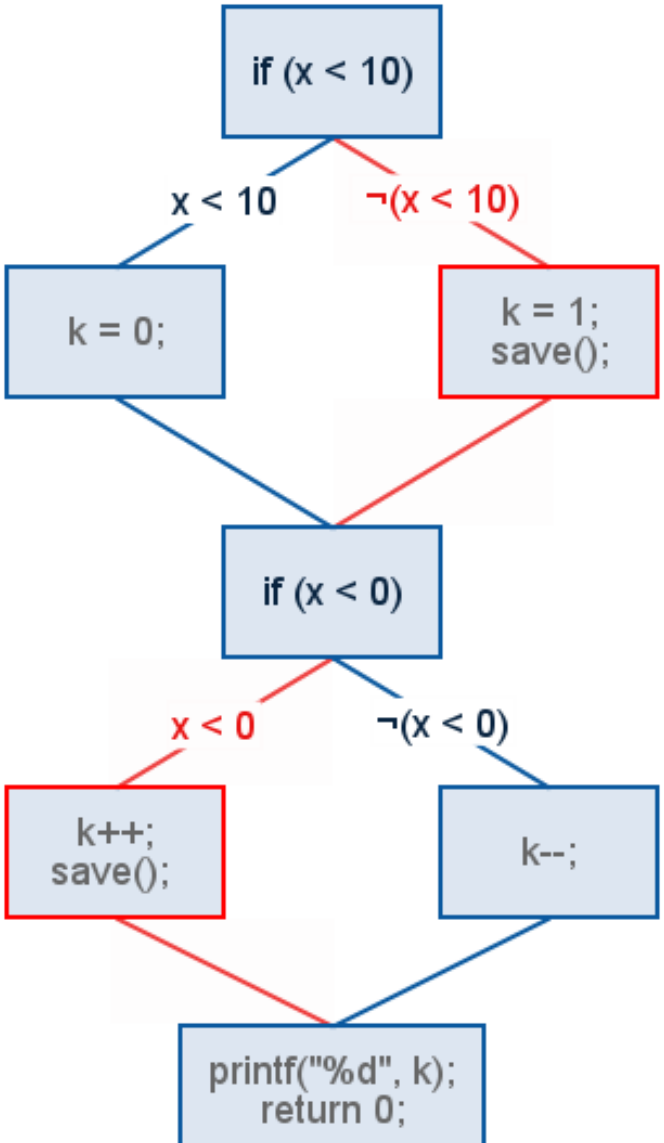
Recherche de chemins infaisables

- Graphe de flot de contrôle



Recherche de chemins infaisables

- Graphe de flot de contrôle
- Chemins + SMT
  - $(x < 10) \wedge (x < 0)$
  - $(x < 10) \wedge \neg(x < 0)$
  - $\neg(x < 10) \wedge (x < 0) \models \bot$
  - $\neg(x < 10) \wedge \neg(x < 0)$

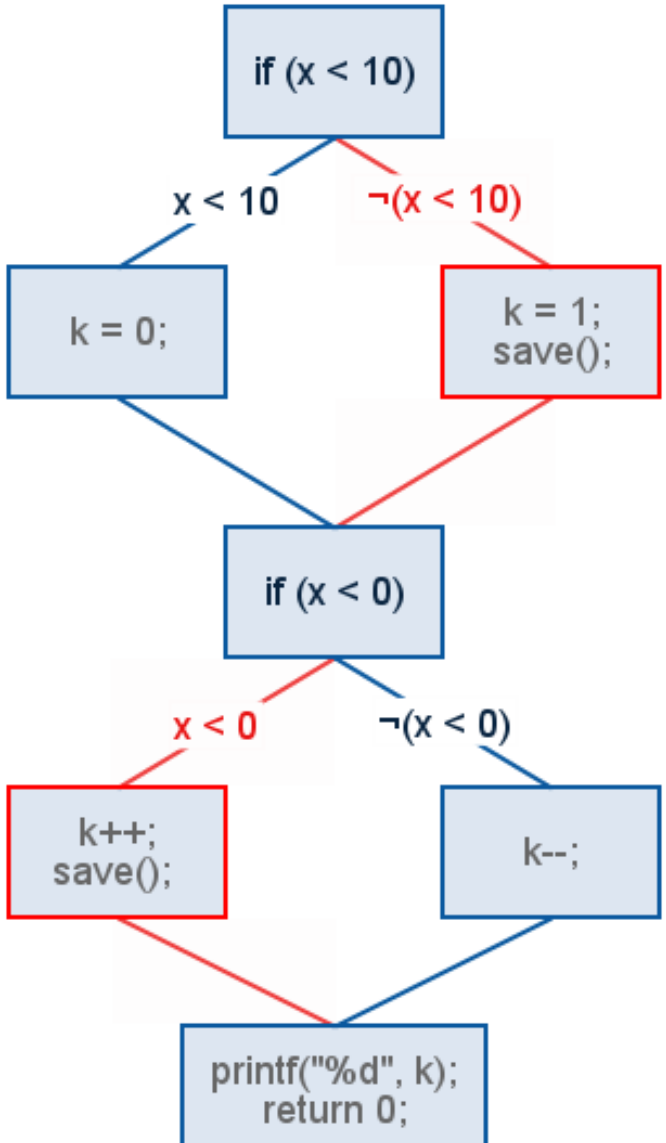


Recherche de chemins infaisables

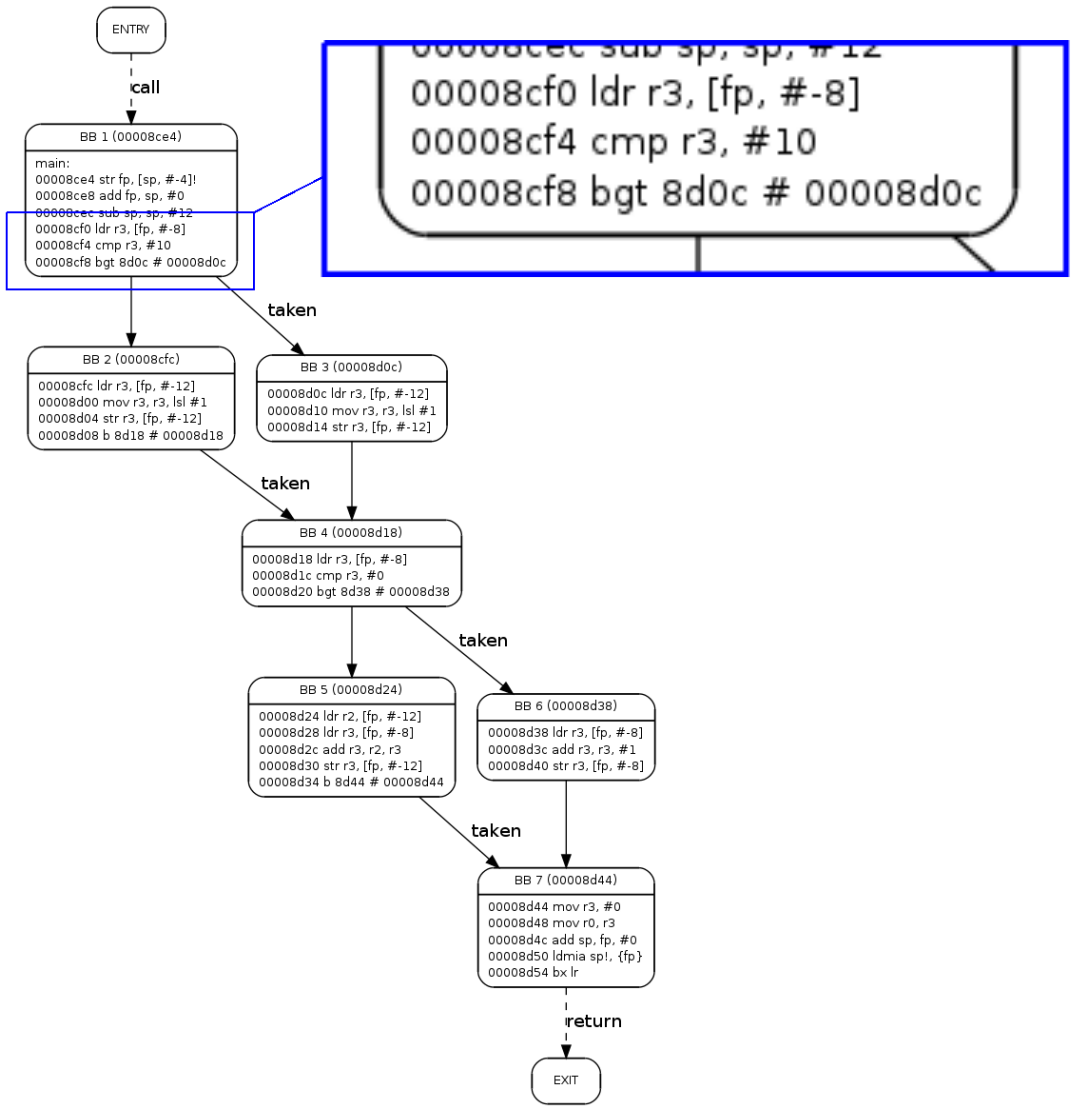
- Graphe de flot de contrôle
- Chemins + SMT
  - $(x < 10) \wedge (x < 0)$
  - $(x < 10) \wedge \neg(x < 0)$
  - $\neg(x < 10) \wedge (x < 0) \models \bot$
  - $\neg(x < 10) \wedge \neg(x < 0)$
- Chemin infaisable

  $\Rightarrow$ contrainte ILP

  $n_{(x<0)} \leq n_{(x<10)}$



Graphe en langage machine



Les instructions sémantiques d'OTAWA

```
ldr r0, [pc, #20]
@ seti ?15, 0x8310
@ seti t2, 0x14
@ add t1, ?15, t2
@ load ?0, t1, uint32

mov r1, #0
@ seti ?1, 0x0

mov r2, r1
@ set t1, ?1
@ set ?2, t1

bl 8574
@ seti t1, 0x8574
@ seti ?14, 0x8318
@ branch t1
```

- Variables temporaires du type t1
- Registres machine du type ?0

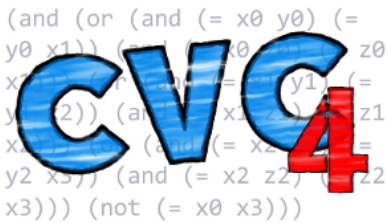
Les instructions sémantiques d'OTAWA

Instruction	Sémantique
`NOP`	(rien)
`BRANCH, TRAP, CONT`	Indicateurs du flot du programme
`IF cond sr jump`	si la condition `cond` sur le registre `sr` est vraie, continuer, sinon sauter `jump` instructions
`LOAD reg addr type`	reg $\leftarrow$ $\texttt{MEM}_{\texttt{type}}$
`STORE reg addr type`	$\texttt{MEM}_{\texttt{type}}$ $\leftarrow$ reg
`SCRATCH d`	`d` $\leftarrow \top$ *(invalidation)*
`SET d a`	`d` $\leftarrow$ `a`
`SETI d cst`	`d` $\leftarrow$ `cst`
`CMP d a b`	`d` $\leftarrow$ `a` $\sim$ `b`
`CMPU d a b`	`d` $\leftarrow$ `a` $\sim_{\texttt{unsigned}}$ `b`
`ADD d a b`	`d` $\leftarrow$ `a` $+$ `b`
`SUB d a b`	`d` $\leftarrow$ `a` $-$ `b`
`SHL d a b`	`d` $\leftarrow$ `unsigned(a) << b`
`SHR d a b`	`d` $\leftarrow$ `unsigned(a) >> b`
`ASR d a b`	`d` $\leftarrow$ `a >> b`
`NEG d a`	`d` $\leftarrow$ $-$`a`
`NOT d a`	`d` $\leftarrow$ $\neg$`a`
`AND d a b`	`d` $\leftarrow$ `a & b`
`OR d a b`	`d` $\leftarrow$ `a | b`
`XOR d a b`	`d` $\leftarrow$ `a` $\oplus$ `b`
`MUL d a b`	`d` $\leftarrow$ `a` $\times$ `b`
`MULU d a b`	`d` $\leftarrow$ `unsigned(a)` $\times$ `unsigned(b)`
`DIV d a b`	`d` $\leftarrow$ `a / b`
`DIVU d a b`	`d` $\leftarrow$ `unsigned(a) / unsigned(b)`
`MOD d a b`	`d` $\leftarrow$ `a % b`
`MODU d a b`	`d` $\leftarrow$ `unsigned(a) % unsigned(b)`

Choix du solveur SMT

Notre choix de solveur s'est porté sur **CVC4** :

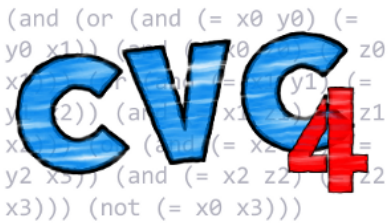
- Open-source, licence très libre



Choix du solveur SMT

Notre choix de solveur s'est porté sur **CVC4** :

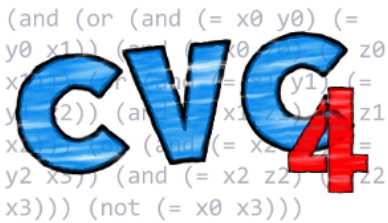
- Open-source, licence très libre
- De très bons résultats à la SMT-COMP



Choix du solveur SMT

Notre choix de solveur s'est porté sur **CVC4** :

- Open-source, licence très libre
- De très bons résultats à la SMT-COMP
- Une API C++ riche et bien documentée



État des travaux

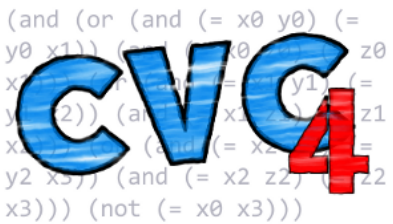
```
   + ?14 = [t1 - 4]
   - ?4 = [(t1 - t2) - t2]
   + ?4 = [(t1 - 4) - 4]
   - ?13 - ((t1 - t2) - t2) = 8
   + ?13 - ((t1 - 4) - 4) = 8
- t2 = 4
- ?14 = [t1 - 4]
- ?4 = [(t1 - 4) - 4]
- ?13 - ((t1 - 4) - 4) = 8
bx lr
branch ?14
Predicates generated: [?0 = 0]
Processing Edge: BB 10 (00008318) -> EXIT (virtual)
EXIT block reached
Processing Edge: BB 5 (000082f4) -> BB 6 (000082fc) (not taken)
SMT call: UNSAT
[Inf. path found: [1->3, 5->6] (bitcode=101)]
Current path identified as infeasible, stopping analysis
Processing Edge: BB 1 (000082d8) -> BB 2 (000082e8) (not taken)
SMT call: SAT
Processing BB 2 (000082e8)
bl 8278
seti t1, 0x8278 (33400)
   + t1 = 0x8278
seti ?14, 0x82ec (33516)
   + ?14 = 0x82ec
branch t1
- t1 = 0x8278
Predicates generated: [?14 = 0x82ec]
Processing Edge: BB 2 (000082e8) -> BB 4 (000082ec) (not taken)
SMT call: SAT
Processing BB 4 (000082ec)
b 82f4
seti t1, 0x82f4 (33524)
   + t1 = 0x82f4
branch t1
- t1 = 0x82f4
```

```
   - (?14 = 0x8318 | 9->10)
   + ?14 = [t1]
add t1, t1, t2
   [t1 - t2 / t1]
      - ?14 = [t1]
      + ?14 = [t1 - t2]
      - ?4 = [t1 - t2]
      + ?4 = [(t1 - t2) - t2]
      - t3 - (t1 - t2) = 8
      + t3 - ((t1 - t2) - t2) = 8
      - t1 - t2 = ?13
      + (t1 - t2) - t2 = ?13
set ?13, t3
   - (t1 - t2) - t2 = ?13
   + ?13 = t3
[?13 / t3]
   - t3 - ((t1 - t2) - t2) = 8
   + ?13 - ((t1 - t2) - t2) = 8
- ?13 = t3
[4 / t2]
   - ?14 = [t1 - t2]
   + ?14 = [t1 - 4]
   - ?4 = [(t1 - t2) - t2]
   + ?4 = [(t1 - 4) - 4]
   - ?13 - ((t1 - t2) - t2) = 8
   + ?13 - ((t1 - 4) - 4) = 8
- t2 = 4
- ?14 = [t1 - 4]
- ?4 = [(t1 - 4) - 4]
- ?13 - ((t1 - 4) - 4) = 8
bx lr
branch ?14
Predicates generated: [?0 = 0]
Processing Edge: BB 10 (00008318) -> EXIT (vir
EXIT block reached
1 infeasible path found:
   - [1->3, 5->6]
```

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
  - Traiter des programmes **avec boucles**
    $\Longrightarrow$ découpage du programme en partie sans boucles (ou avec boucles simples)

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
  - Traiter des programmes **avec boucles**
    $\Longrightarrow$ découpage du programme en partie sans boucles (ou avec boucles simples)
  - Appels au solveur SMT plus intelligents



- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
  - Traiter des programmes **avec boucles**
    $\Longrightarrow$ découpage du programme en partie sans boucles (ou avec boucles simples)
  - Appels au solveur SMT plus intelligents
  - Gérer les spécificités des types de données du langage machine

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
  - Traiter des programmes **avec boucles**
    $\Longrightarrow$ découpage du programme en partie sans boucles (ou avec boucles simples)
  - Appels au solveur SMT plus intelligents
  - Gérer les spécificités des types de données du langage machine
  - Générer des contraintes ILP ne **suffit plus**
    $\Longrightarrow$ il faudrait faire de la **réécriture de graphe**.

Conclusion

- La recherche de chemins infaisables : un problème d'actualité


Questions ?