

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

Candidat
Jordy RUIZ

Encadrant
Hugues CASSÉ

19 juin 2014

Sommaire

- 1 Introduction
- 2 Problématique et contexte
- 3 Sujet de thèse proposé
- 4 Conclusion

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)

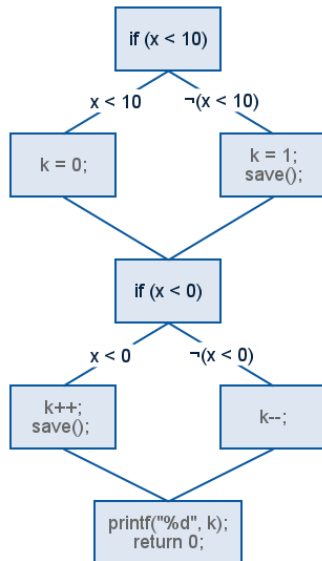
- But : surestimer le temps d'exécution d'une partie de programme
- Exemples
 - Le frein de voiture s'activera au pire 50ms après la commande
 - L'algorithme prendra une décision en moins d'1s
 - ...
- Systèmes temps-réel critiques : les mesures ne suffisent pas, il faut une **preuve** !

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

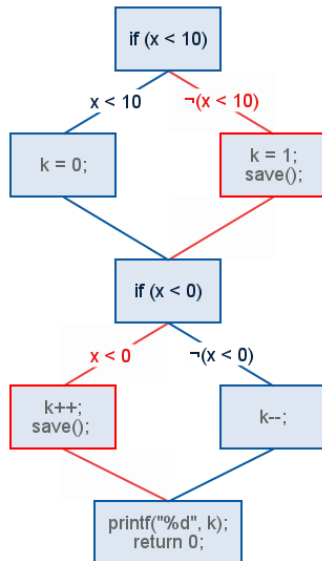
Recherche de chemins infaisables

- Graphe de flot de contrôle



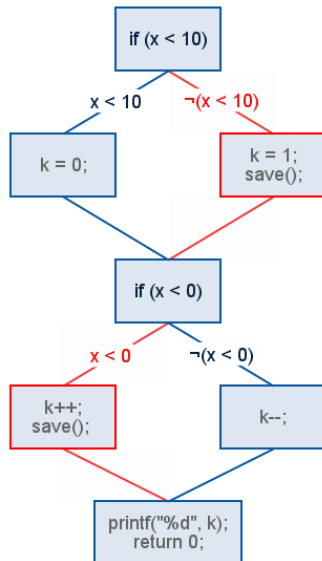
Recherche de chemins infaisables

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

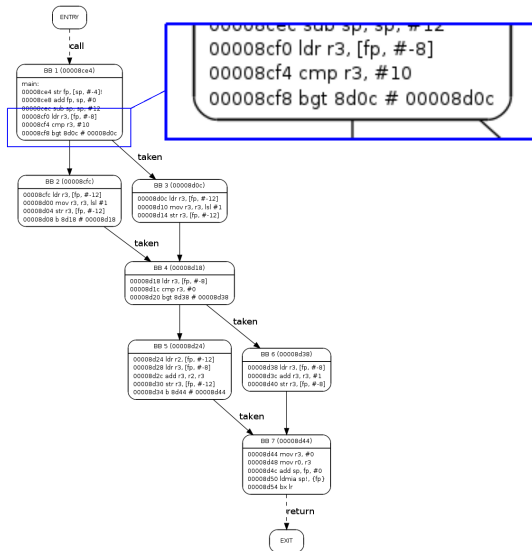


Recherche de chemins infaisables

- Graphe de flot de contrôle
- Chemins + SMT
 - $(x < 10) \wedge (x < 0)$
 - $(x < 10) \wedge \neg(x < 0)$
 - $\neg(x < 10) \wedge (x < 0) \models \perp$
 - $\neg(x < 10) \wedge \neg(x < 0)$
- 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	$reg \leftarrow MEM_{type}$
STORE reg addr type	$MEM_{type} \leftarrow reg$
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	$d \leftarrow a \sim_{unsigned} b$
ADD d a b	$d \leftarrow a + b$
SUB d a b	$d \leftarrow a - b$
SHL d a b	$d \leftarrow unsigned(a) \ll b$
SHR d a b	$d \leftarrow unsigned(a) \gg b$
ASR d a b	$d \leftarrow a \gg b$
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

```

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

```


9] seti ?14, 0x82ec
417] - (?14 = 0x8318 | 9->10)
388] + ?14 = [t1]
30] add t1, t1, t2
316] [t1 - t2 / t1]
345] - ?14 = [t1]
353] + ?14 = [t1 - t2]
345] - ?4 = [t1 - t2]
353] + ?4 = [(t1 - t2) - t2]
345] - t3 - (t1 - t2) = 8
353] + t3 - ((t1 - t2) - t2) = 8
345] - t1 - t2 = ?13
353] + (t1 - t2) - t2 = ?13
30] set ?13, t3
431] - (t1 - t2) - t2 = ?13
388] + ?13 = t3
300] [?13 / t3]
304] - t3 - ((t1 - t2) - t2) = 8
305] + ?13 - ((t1 - t2) - t2) = 8
460] - ?13 = t3
300] [4 / t2]
304] - ?14 = [t1 - t2]
305] + ?14 = [t1 - 4]
304] - ?4 = [(t1 - t2) - t2]
305] + ?4 = [(t1 - 4) - 4]
304] - ?13 - ((t1 - t2) - t2) = 8
305] + ?13 - ((t1 - 4) - 4) = 8
460] - t2 = 4
474] - ?14 = [t1 - 4]
474] - ?4 = [(t1 - 4) - 4]
474] - ?13 - ((t1 - 4) - 4) = 8
22] bx lr
30] branch ?14
403] Predicates generated: [?0 = 0]
402] Processing Edge: BB 10 (00008318) -> EXIT (virtual)
352] EXIT block reached
29] 1 infeasible path found:
44] - [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**
⇒ 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**
⇒ découpage du programme en partie sans boucles (ou avec boucles simples)
 - Appels au solveur SMT plus intelligents



The image shows the CVC4 logo, which consists of the letters 'CVC' in a large, blue, stylized font with a 3D effect, and a red '4' to its right. The logo is superimposed on a background of SMT (Satisfiability Modulo Theories) logic code. The visible code includes expressions like '(and (or (and (= x0 y0) (= y0 x1)) (and (= x0 z0) (= x1 z0)))', '(and (= x2 z1) (= x3 z1))', '(and (= x2 z2) (= x3 z2))', and '(not (= x0 x3))'.

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
 - Traiter des programmes **avec boucles**
⇒ 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

```
x = (unsigned int) y;
```

- Thèse dans la continuation du stage M2R
- Problème difficile, très restreint pour le stage. Beaucoup d'extensions à faire :
 - Traiter des programmes **avec boucles**
⇒ 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**
⇒ il faudrait faire de la **réécriture de graphe**.

Conclusion

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

