



Figure 1: OTAWA permet aux analyses de s'abstraire de l'architecture

1 Brouillon

2 Objectif

2.1 WCET, enjeux (systèmes critiques etc)

Améliorer l'estimation de WCET

3 Solution

Explication : calcul de WCET par IPET, max d'un système ILP

Solution = Recherche de chemins infaisables pour améliorer l'estimation du WCET

Parler des outils choisis (OTAWA + CVC4 ?)

4 Structures de données utilisées

La représentation des prédicats (associés à un Edge)

=> représentation également de la mémoire

Traduction des prédicats dans le solveur SMT

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	$\text{reg} \leftarrow \text{MEM}_{\text{type}}$
STORE reg addr type	$\text{MEM}_{\text{type}} \leftarrow \text{reg}$
SCRATCH d	$d \leftarrow \top$ (<i>invalidation</i>)
SET d a	$d \leftarrow a$
SETI d cst	$d \leftarrow \text{cst}$
SETP d cst	$\text{page}(d) \leftarrow \text{cst}$
CMP d a b	$d \leftarrow a \sim b$
CMPU d a b	$d \leftarrow a \sim_{\text{unsigned}} b$
ADD d a b	$d \leftarrow a + b$
SUB d a b	$d \leftarrow a - b$
SHL d a b	$d \leftarrow \text{unsigned}(a) \ll b$
SHR d a b	$d \leftarrow \text{unsigned}(a) \gg b$
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	$d \leftarrow \text{unsigned}(a) \times \text{unsigned}(b)$
DIV d a b	$d \leftarrow a / b$
DIVU d a b	$d \leftarrow \text{unsigned}(a) / \text{unsigned}(b)$
MOD d a b	$d \leftarrow a \% b$
MODU d a b	$d \leftarrow \text{unsigned}(a) \% \text{unsigned}(b)$
SPEC	(instruction spéciale non supportée par OTAWA)

En gris : les instructions qui ne sont pas (encore) traitées par notre analyse.

Figure 2: Liste des instructions sémantiques d'OTAWA

5 Structure de l'algorithme

CFG : Représentation du programme sous la forme de graphe

On parcourt tous les chemins

On les représente sous la forme de prédicats (on y associe un arc du CFG)

On fait des appels au SMT pour vérifier la satisfiabilité

Retourne la liste de prédicats

On génère les contraintes ILP