Identifying Relevant Parameters to Improve WCET Analysis

Jakob Zwirchmayr[2], Pascal Sotin[2], Armelle Bonenfant[2]
Denis Claraz[3], Philippe Cuenot[3]

WCET Workshop – Madrid, July 8, 2014

[1]This work is supported by ANR W-SEPT.
[2]Université de Toulouse
[3]Continental Automotive

Introduction

- W-SEPT: eliminate sources of imprecision on all levels
- highly configurable systems (“product platforms”)

Industrial Context

- collaboration with Continental Automotive
- WCET analysis of a software module

are our tools appropriate?

- analysis during design phase (size hardware)
- analysis during development (compliance)
- analysis of configurations/parameters

WCET Context

- OTAWA toolbox
- oRange analyzer

requires

- module harness (run)
- stubs for dependencies (compilation)

WCET Context

- WCET analysis: valid (safe) estimate
- problem: not really useful

high overestimation due to unspecified configuration

- configuration/parameters = “scenario”

Infeasible Paths due to Scenarios

- configuration can be specified in harness
- "disables" some (originally valid) execution paths
- $\rightarrow$ more precise estimate for the config

scenarios

- expert knows a (large) set of parameters/configuration variables
- for each, the expert specifies possible values
- specification is tedious

Goal

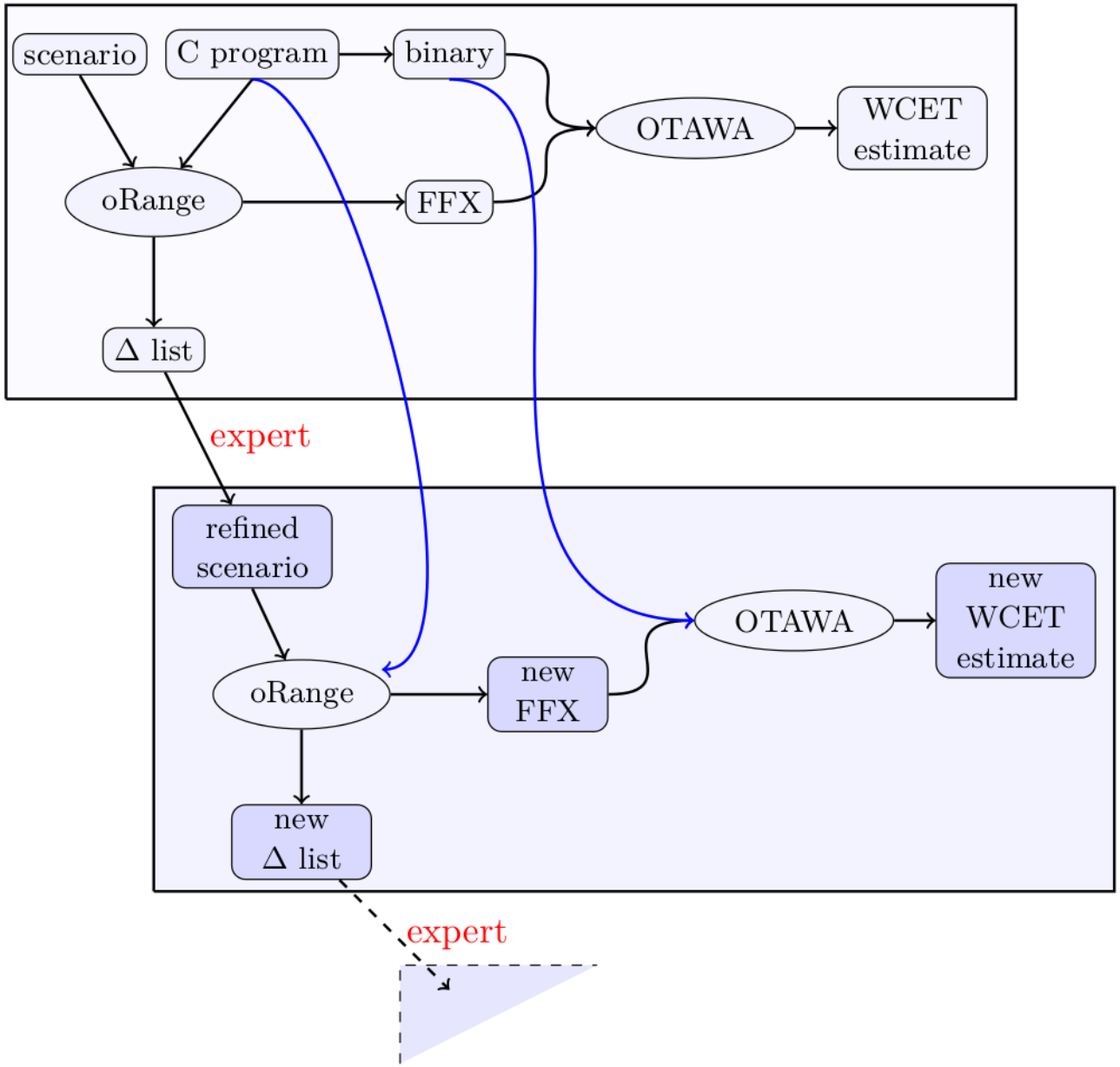
automatic selection of “relevant” variables

- “relevant” variables: enable or disable “heavy” branches
- “light” branches don’t matter
- but are similarly tedious to specify

infer relation between parameter variables and “heavy” branches

- “heavy”: branch is much more expensive than alternative
- ask expert for specification of only those
- compute a more precise estimate for the config

Workflow


scenario
C program
binary
OTAWA
WCET estimate
oRange
FFX
Δ list
expert
refined scenario
OTAWA
new WCET estimate
oRange
new FFX
new Δ list
expert

Workflow

- identify relevant parameter variables from the set
- get additional constraints/assumptions only on those (expert)
- WCET analyze the module under the given assumptions

$\rightarrow$ profit from additional specification
$\rightarrow$ keep specification effort low
(no need to identify relevant parameters manually)

Principle of the Analysis

- source-based analysis to identify “unbalanced” conditionals
- simple matching with set of parameters
- use spec of those, if possible without modifying the harness

- branching statement analysis
- uses information inferred from high-level analyzer oRange (loopbounds, executed)
- outputs a selection of parameter variables
- encode them as input annotations to oRange
- exploit (output) flow facts in OTAWA

Branching Statement Analysis

- source-based, simple cost model (can be updated)
- on (internal) oRange representation (AST)
- weight computed bottom up from AST nodes

- input: C program
- output: list of locations, branches, Δ-value and condition
- Δ-value: indicates "unbalancedness" of branches

- match Δ-conditions to parameters

Output of the Analysis

```
#include "missing.h "
int main () {
  for (int i = 0; i < 100; i ++)
    if (max_speed > 250) expensive ();
    else cheap();

Computing the balance information for the main function
Estimated cost of the function : 70804
1 accessible conditional statements
Delta 65000 at rex.c :22 in main (total count = 100) :
then = 704; else = 54; // max_speed > 250
```

Eliminating Infeasible Branches

- eliminating infeasible branches on the WCETP improves estimate
- “heavy” v.s. “light”
- if not on the WCETP, improvement is possible as well

no improvement:

- “witness for accuracy”
- scenario-WCET coincides with WCET

Continental Use-Case

- module of 700 loC
- list of 85 possible parameters (existing)
- scenario specifying 30 parameters (expert chosen)

Results

scenario	# parameters	no cache		cache	
(1) global, no scenario	0	2553	gain	6883	gain
(2) full scenario	30	2426	5%	6486	5.7%
(3) 3 highest Δ	3	2553	0%	6833	0%
(4) 8 highest Δ	10	2479	3%	6679	3%
(5) 9 highest Δ	14	2463	3.5%	6623	3.8%
(6) 10 highest Δ	18	2448	4%	6568	4.6%
(7) inverted 3 highest Δ	3 (inverted)	2055	19%	5795	15.8%
(8) none of Δ	10	2551	0.08%	6831	0.03%

- scenario exec-t. coincides with WCET, nevertheless:
- relevant parameters are identified: 20 of 30 in Δ-conds
- matches 18 parameters in 10 highest Δ
- parameters not listed in Δ-conds have "no relevance" (0.3%)
- **3 most relevant variables have high impact**

Conclusion

- BSA does indeed identify relevant parameters
- exploiting them in scenarios is possible

- can be applied early, hinting at relevant parameters
- decreases/eliminates manual effort to identifying them

Future Work

- more realistic cost models
- matching conditions $\leftrightarrow$ parameters (dependencies)

- richer input annotation (intervals, relations)
- exploiting additional information in OTAWA (e.g. executed)

- combine with counter-based program analysis techniques (e.g. identify important counter locations)