New challenges in adaptive real-time systems with parametric WCET

Clément Ballabriga, **Julien Forget**, Sandro Grebant, Giuseppe Lipari

Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, Lille, France
`mailto: julien.forget@univ-lille.fr`

RTSOPS'23 - July 11, 2023

Outline

Parametric WCET analysis

- Classical WCET analysis:
  - Compute a **single numeric value**;
  - Upper-bound for all software/hardware parameter combinations;
  - Often largely pessimistic;
- Parametric WCET analysis:
  - Compute a **formula** of software/hardware parameters;
  - **Instantiate** formula when parameter values become known.

Example

WCET variability

```
void f(int a){
  // 10
  if(a > 0)
    // 15
  else
    // 5
  //10
}
```

$$WCET = 10 + max(15, 5) + 10 = 35$$

Example

WCET variability

```
void f(int a){
  // 10
  if(a > 0)
    // 15
  else
    // 5
  //10
}
```

$$WCET = \begin{cases} 10 + 15 + 10 = 35 & \textit{if } a > 0 \\ 10 + 5 + 10 = 25 & \textit{otherwise} \end{cases}$$

Contribution

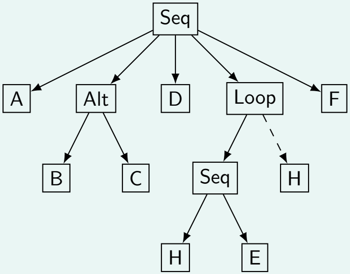
- Comparison with other parametric WCET approaches:
  - More parameter kinds;
  - Adaptive;
  - Embeddable;
  - Automated.
- **Applications**:
  - Off-line instantiation: parameter-space exploration;
  - On-line instantiation: adaptive system.

Outline

Tree-based WCET

Control-Flow Tree

```
void f(int a, int b){
  // A
  if(a > 0)
    // B
  else
    // C
  // D
  for(int i=0; i<a+b; i++) // H
    // E
  // F
}
```



Compute WCET recursively on the tree:

- *Seq* = addition;
- *Alt* = max;
- *Loop* = multiply by max iterations.

Problem: context-dependent WCET

- Execution time of a node often depends on its execution context;
- Easily represented in IPET (more ILP constraints);
- Not captured by the tree structure.

Example: first-miss

Cache timing effect when iterating a node:

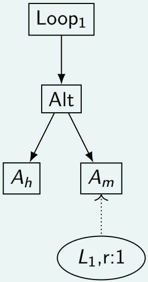
- Miss on first iteration $\Rightarrow$ higher execution time
- Hit on other iterations $\Rightarrow$ lower execution time.

Context annotations [Ballabriga et al., 2017]

Context annotation $(L_n, r : n)$:

- For a complete execution of loop L_n...
- ...annotated node is executed at most n times.

First-miss



WCET of a node

Context annotations ⇒ WCET is a list of values:

- Non-increasing order;
- Smallest element implicitly repeated infinitely;
- Specify which loop is taken as reference.

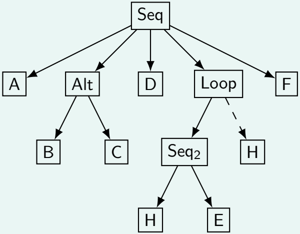
Cache first-miss

- Assume:
  - $\omega(A_m) = (L_1, [25])$
  - $\omega(A_h) = (L_1, [15])$
  - A_m annotated with $(L_1, r : 1)$;
- Then:
  - $\omega(Alt(A_h, A_m)) = (L_1, [25, 15])$
  - When iterating inside L_1, WCET is 25, 15, 15, 15, ...

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula

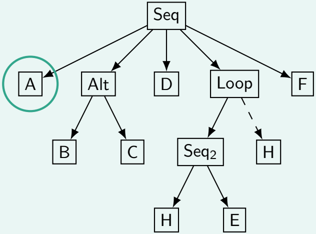


$\omega(\text{Seq}) =$

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula



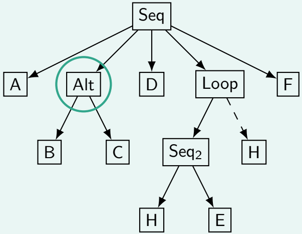
$\omega(\text{Seq}) =$

$\omega(A)$

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula



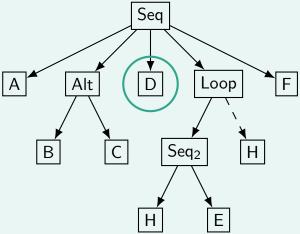
$\omega(\text{Seq}) =$

$$\omega(A) \oplus (\omega(B) \uplus \omega(C))$$

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula



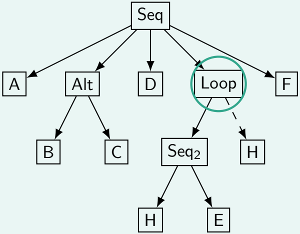
$\omega(\text{Seq}) =$

$$\omega(A) \oplus (\omega(B) \uplus \omega(C)) \oplus \omega(D)$$

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula



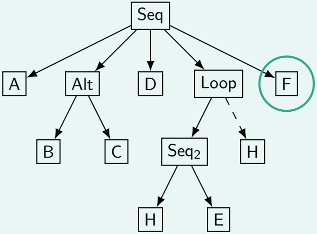
$\omega(\text{Seq}) =$

$$\omega(A) \oplus (\omega(B) \uplus \omega(C)) \oplus \omega(D) \oplus (\omega(H) \oplus \omega(E))^n \oplus \omega(H)$$

WCET computation

- WCET formula computed inductively on the tree structure;
- Operations on WCET lists:
  - $w_1 \oplus w_2$: point-wise sum;
  - $w_1 \uplus w_2$: list union;
  - w^n: sum values of w by packs of n.

WCET formula



$\omega(\text{Seq}) =$

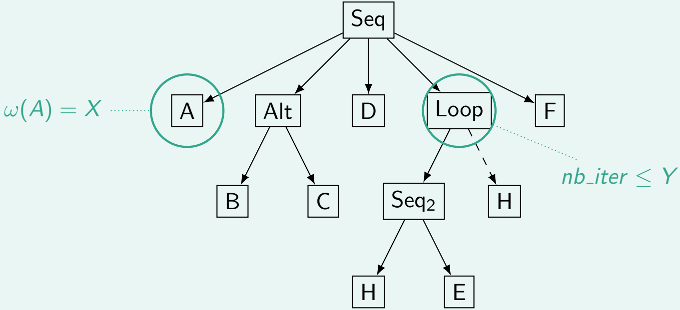
$$\omega(A) \oplus (\omega(B) \uplus \omega(C)) \oplus \omega(D) \oplus (\omega(H) \oplus \omega(E))^n \oplus \omega(H) \oplus \omega(F)$$

Symbolic values

Some elements of the CFT can be unknown, a.k.a symbolic:

- Node with a symbolic WCET;
- Symbolic loop bounds.

CFT with parameters



Procedure arguments as parameters [Grebant et al., 2023]

Step 1: Infer branch conditions by **relational** abstract interpretation.

Inferring input conditionals

```
f:                              @ void f(int a, int b){
  @ ...                         @
  str  r0, [fp, #-16]           @   // r0 contains arg1
  str  r1, [fp, #-20]           @   // r1 contains arg2
  @ ...                         @
  ldr  r3, [fp, #-16]           @
  cmp  r3, #0                   @
  ble  .L2                      @   if(a > 0) {
  @ ...                         @
  b   .L3                       @   }
.L2:                            @   else {
  @...                          @
.L3:                            @   }
  @ ...                         @   // ...
```

Procedure arguments as parameters [Grebant et al., 2023]

Step 1: Infer branch conditions by **relational** abstract interpretation.

Inferring input conditionals

```
f:                              @  void f(int a, int b){
  @ ...                         @
  str  r0, [fp, #-16]           @    // r0 contains arg1
  str  r1, [fp, #-20]           @    // r1 contains arg2
  @ ...                         @
  ldr  r3, [fp, #-16]           @
  cmp  r3, #0                   @    ▷ Test on arg_1
  ble  .L2                      @    if(a > 0) { ▷ not obvious in assembly
  @ ...                         @
  b    .L3                      @    }
.L2:                            @    else {
  @...                          @
.L3:                            @    }
  @ ...                         @    // ...
```

Procedure arguments as parameters [Grebant et al., 2023]

Step 1: Infer branch conditions by **relational** abstract interpretation.

Inferring input conditionals

```
f:                                  @  void f(int a, int b){
  @ ...                             @
  str   r0, [fp, #-16]              @    // r0 contains arg1
  str   r1, [fp, #-20]              @    // r1 contains arg2
  @ ...                             @
  ldr   r3, [fp, #-16]              @
  cmp   r3, #0                      @    ▷ Test on arg_1
  ble   .L2                         @    if(a > 0) { ▷ not obvious in assembly
  @ ...                             @      ▷ "then" condition: arg_1 > 0
  b   .L3                           @    }
.L2:                                @    else {
  @...                              @
.L3:                                @    }
  @ ...                             @    // ...
```

Procedure arguments as parameters [Grebant et al., 2023]

Step 1: Infer branch conditions by **relational** abstract interpretation.

Inferring input conditionals

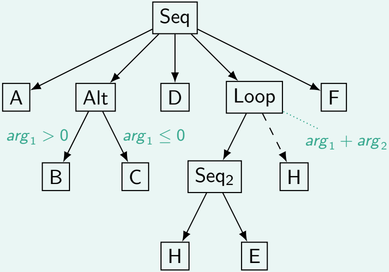
```
f:                              @ void f(int a, int b){
  @ ...                         @
  str  r0, [fp, #-16]           @   // r0 contains arg1
  str  r1, [fp, #-20]           @   // r1 contains arg2
  @ ...                         @
  ldr  r3, [fp, #-16]           @
  cmp  r3, #0                   @   ▷ Test on arg_1
  ble  .L2                      @   if(a > 0) { ▷ not obvious in assembly
  @ ...                         @     ▷ "then" condition: arg_1 > 0
  b    .L3                      @   }
.L2:                            @   else {
  @...                          @     ▷ "else" condition: arg_1 ≤ 0
.L3:                            @   }
  @ ...                         @   // ...
```

Procedure arguments as parameters

Step2: Insert AI results in the CFT.

- Condition of an alternative: conjunction of inequations on arguments;
- Loop bound: linear expression on arguments.

Input conditionals in loops



Formula simplification

- CFT with symbolic values $\Rightarrow$ formula not reducible to a WCET list;
- Simplify formula based on algebraic properties:
  - Define custom rewriting rules $l \mapsto r$;
  - Prove $l \Leftrightarrow r$ for each rule;
  - Repeatedly apply rewriting rules;
- Simplified formula compiled to C code $\Rightarrow$ on-line instantiation.

Example

$$f = ((a > 0) \circledast (l, [10, 5])) \uplus ((a \leq 0) \circledast (l, [10, 5]))$$
$$= (l, [10, 5]) \qquad \text{(Since } a > 0 \Leftrightarrow \neg(a \leq 0))$$

Outline

Input-dependent WCET formulas

WCET formulas vs other models with several WCET values:

1. Explicit WCET-to-inputs dependence;
2. A single input might impact the WCET of several tasks;
3. Not an obvious number of different WCET values for a task.

Open problem 1

Sensitivity analysis

Which input values make a task set schedulable?

Main difficulty:

(2) **A single input might impact the WCET of several tasks.**

Open problem 2

Semi-clairvoyant scheduling

Semi-clairvoyant scheduling of tasks with WCET formulas.

Main difficulty:

(3) **Not an obvious number of different WCET values.**

Conclusion

Symbolic WCET computation:

- Embeddable;
- Adaptive;
- Automated.

Paving the way for new kinds of adaptive real-time systems?

Downloads

- Polymalys (AI): https://gitlab.cristal.univ-lille.fr/otawa-plugins/polymalys
- WSymb (WCET): https://gitlab.cristal.univ-lille.fr/otawa-plugins/WSymb
- RTNS'23 artifact: https://gitlab.cristal.univ-lille.fr/sgrebant/rtns_2023_artifact

References

[Ballabriga et al., 2017] Ballabriga, C., Forget, J., and Lipari, G. (2017).
Symbolic WCET Computation.
ACM Transactions on Embedded Computing Systems (TECS), 17(2):1 – 26.

[Grebant et al., 2023] Grebant, S., Ballabriga, C., Forget, J., and Lipari, G. (2023).
WCET analysis with procedure arguments as parameters.
In *RTNS 2023: The 31st International Conference on Real-Time Networks and Systems*.