UMR IRISA

The Heptane Static Worst-Case Execution Time Estimation Tool

Damien Hardy
damien.hardy@irisa.fr

Benjamin Rouxel
benjamin.rouxel@irisa.fr

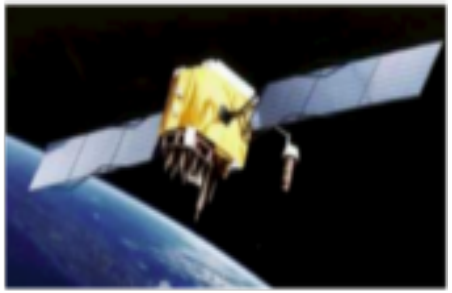
Isabelle Puaut
isabelle.puaut@irisa.fr

Institut de Recherche en Informatique et Systèmes Aléatoires

Introduction

Real-time system
Timing guarantees
Worst-Case Execution Time (WCET)

- Static methods
- Measurement-based
- Probabilistic



Heptane: a static analysis tool

- Program structure analysis
- Flow information
- Binary level
- Model of the architecture

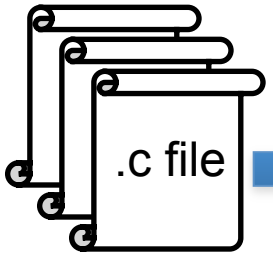
About HEPTANE

- Developed at IRISA/University of Rennes
  - Stable research prototype
  - Licenced under GPL
  - C++ code
  - Supported architectures
    - MIPS
    - Partially: ARM v7, Patmos (under construction)

- The first version has been designed around 2000
  - Many updates since then!
  - Available online

https://team.inria.fr/pacap/software/heptane/

Heptane toolchain



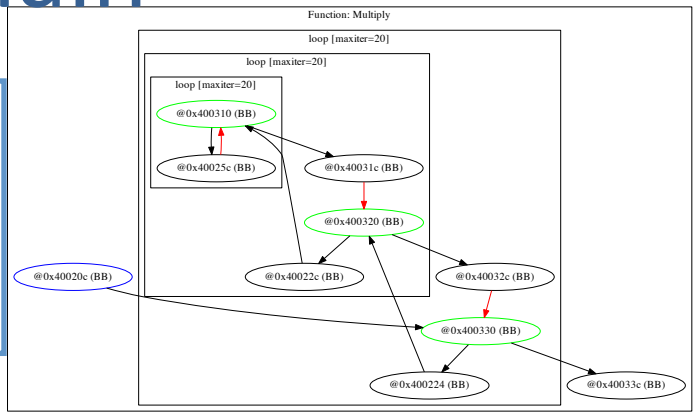
& loops bounds

Compiler (gcc/LLVM)

Binary

CFG extraction

HeptaneExtract



XML file

HeptaneAnalysis

Low-Level Analysis

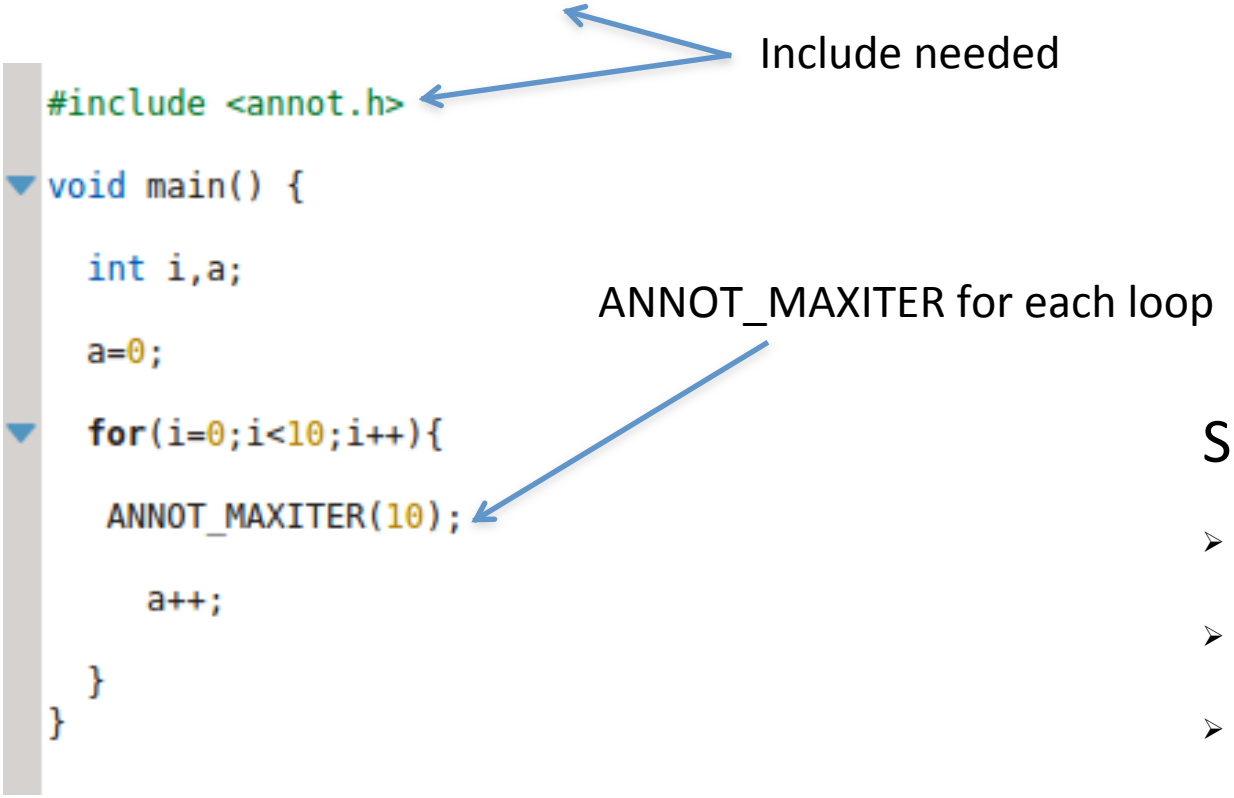
High-Level Analysis

```
0                   register int    Outer, Inner, Index;
0
1 20 21             for (Outer = 0; Outer < UPPERLIMIT; Outer++) {
0                     ANNOT_MAXITER(20);
20 400 420            for (Inner = 0; Inner < UPPERLIMIT; Inner++) {
0                       ANNOT_MAXITER(20);
```

- WCET
- Statistics
- HTML

How to use it

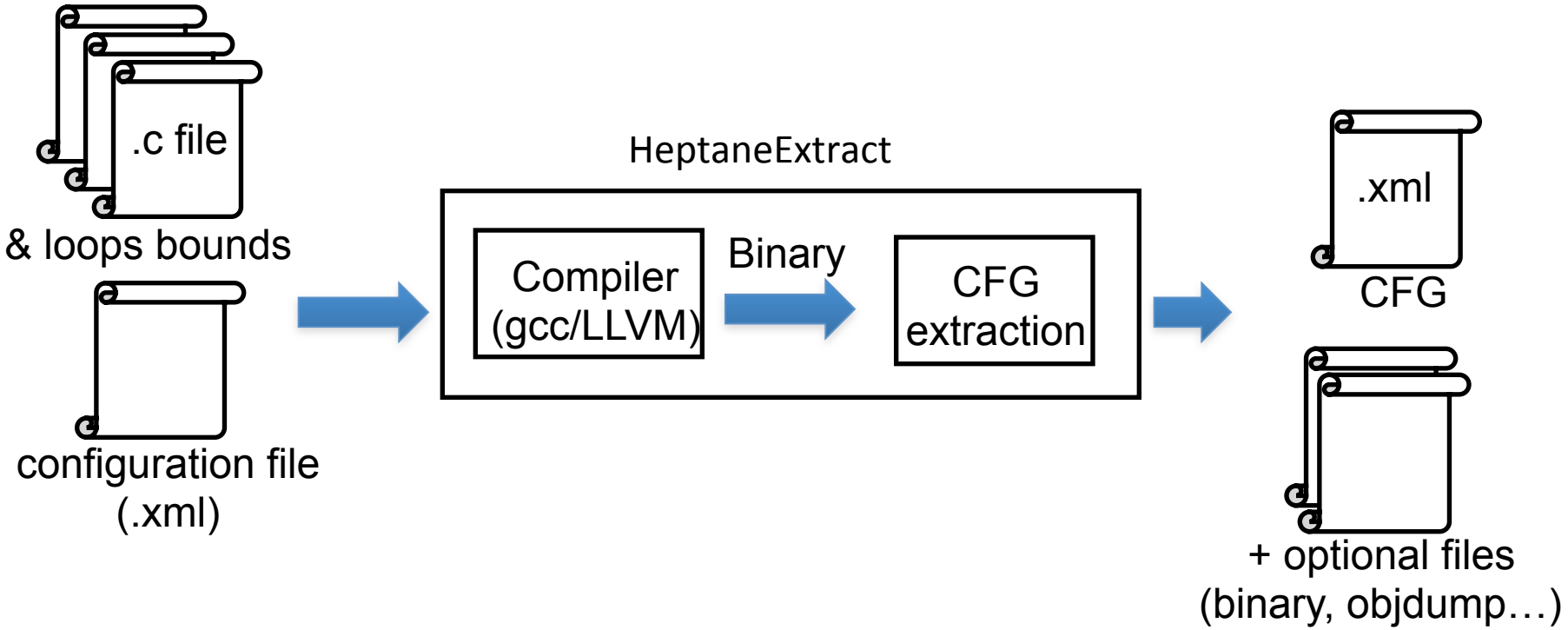
- Source code in C (alternatively binary)
  - With annotation for the loops bounds: ANNOT_MAXITER(cst)



Some restrictions :

- No pointers
- No malloc
- No recursion
- Natural loops

How to use it: HeptaneExtract



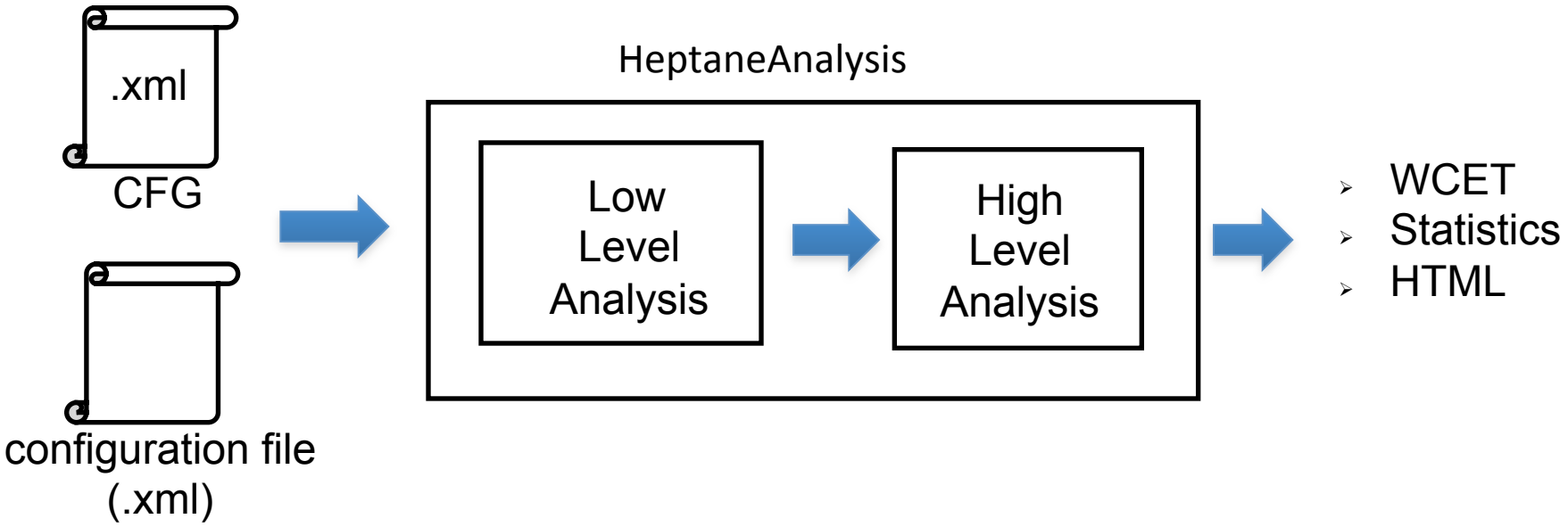
CFG.xml overview

```
<!DOCTYPE PROGRAM SYSTEM 'cfglib.dtd'>
<PROGRAM   id ="1"   name="simple"   entry="0" >
<CFG id="0"  name="main"  startnode="2"  endnodes="3, " >
  <NODE  id="2"  type="BasicBlock" >
  <INSTRUCTION  id="4"  asm_type="Code"  code="addiu sp,sp,-8"  >
<ATTRS_LIST>
<ATTR type="address" name="address">
  <ACCES type="read" seg="code" varname="" precision="1">
      <ADDRSIZE  begin="0x400018" size="4"/>
  </ACCES>
</ATTR>
</ATTRS_LIST>
  </INSTRUCTION>
  <INSTRUCTION  id="5"  asm_type="Code"  code="sw zero,4(sp)"  >
<ATTRS_LIST>
<ATTR type="address" name="address">
  <ACCES type="read" seg="code" varname="" precision="1">
      <ADDRSIZE  begin="0x40001c" size="4"/>
  </ACCES>
</ATTR>
</ATTRS_LIST>
  </INSTRUCTION>
```

```
    <EDGE  id="25"  origin="2"  destination="17" >
    </EDGE>

    <EDGE  id="26"  origin="8"  destination="17" >
    </EDGE>

    <EDGE  id="27"  origin="17"  destination="8" >
    </EDGE>

    <EDGE  id="28"  origin="17"  destination="3" >
    </EDGE>

<LOOP  id="29"  head="17"  nodes="17, 8, "  backedges="26, " >
<ATTRS_LIST>
<ATTR type="integer" name="maxiter" value="10" />
</ATTRS_LIST>
</LOOP>
```

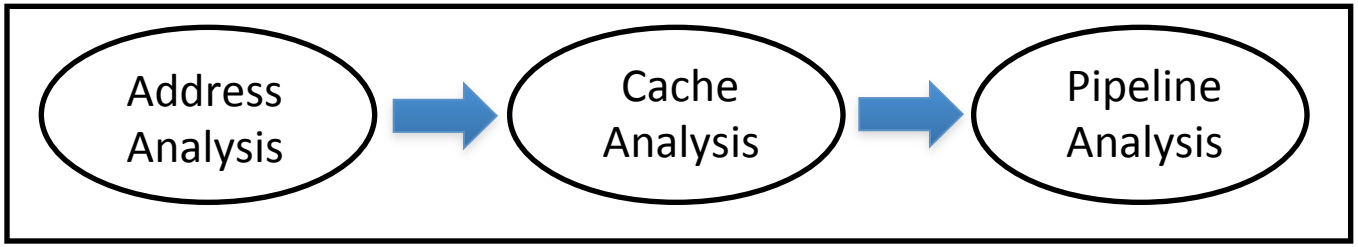
- 1 cfg per function
- Instruction listing filling basic-blocks
- Instruction addresses
- Edges between basic-blocks
- Attributes (e.g. loop bounds)

How to use it: HeptaneAnalysis



Low-level analysis

- Account for micro-architecture
- Timing information per basic-block
- Enrich the CFG.xml



Data address analysis

- Conservatively determines addresses of referenced data
- Instruction addresses from the binary
- Stack analysis per function
  - first instruction of functions
  - calling context
- Basic block granularity
  - for each access (load/store)
  - evaluate registers content + memory (new)

Cache analysis

- Instruction & data set-associative caches
  - contextual analysis
  - abstract interpretation (must, may, persistence)
- Multi-level non inclusive cache hierarchies
  - write-through, no-write allocate data caches
  - LRU, PLRU, FIFO, MRU, Random
- Cache Hit Miss Classification (CHMC)
  - Always hit, Always miss, First miss, Not-classified

Pipeline analysis

- Compute the timing of basic blocks
  - simple in-order pipeline
  - free of timing anomalies
  - fetch/memory stage depends on cache analysis

High-level analysis

- Determine critical path
- Context sensitive (calling context)
- Implicit Path Enumeration Technique (IPET)

maximize $\sum_{i \in BasicBlocks} T_i \times f_i$

T_i: execution time, constant computed by low-level analysis

f_i: # executions

Install, use and hack

Install

GNU/Linux & MacOS

download sources from : https://team.inria.fr/pacap/software/heptane/
modify *install.sh* to your configuration
dependencies: libxml2, ILP solver (lp-solve, cplex)

Use

create a config file (sample in *config_files/* folder)
launch *HeptaneExtract*
launch *HeptaneAnalysis*
collect results

Hack

architecture specificities in *src/Common/ArchitectureDependent/*
analysis in *src/HeptaneAnalysis/src/Specific/*

Research based on Heptane

- Cache analysis (cache hierarchies, multi-cores)

D. Hardy, I. Puaut. WCET analysis of instruction cache hierarchies. Journal of Systems Architecture, volume 57, issue 7, pages 677-694, August 2011.

- Branch prediction (analysis, compiler-directed)

A. Colin, I. Puaut. Worst Case Execution Time Analysis for a Processor with Branch Prediction. Real-Time Systems, April 2000.

- Cache locking, partitioning

I. Puaut. Cache Analysis vs Static Cache Locking for Schedulability Analysis in Multitasking Real-Time Systems. WCET Workshop, ECRTS 2002.

- Scratchpad management

J-F. Deverge, I. Puaut. WCET-directed dynamic scratchpad memory allocation of data. ECRTS 2007.

- CRPD estimation, Static probabilistic WCET, Traceability of flow information, ...

https://team.inria.fr/pacap/members/isabelle-puaut/

Conclusion

Heptane WCET analysis tool

- open source
- estimate WCET from binary
- advanced in static cache analysis

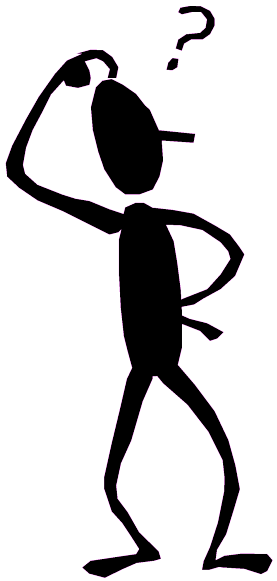
Side product : CFG management library

Future work

- automatic loop bound extraction
- further improve address analysis
- increase number of supported architecture

Any help is welcome

Questions?



https://team.inria.fr/pacap/software/heptane/

Specific questions, skype: isabellepuaut