TACLe
Timing Analysis on Code-Level

WCET Tool Challenge 2014

Outline

1. Objectives of the challenge
2. Benchmarks and problems
3. Participants
4. State of progress / next?

Objectives

- a set of benchmarks and problems to:
  - compare the properties of different **WCET tools**
    - what is the state of the art?
    - friendly competition, no winner!
  - improve **benchmarks** and collect new ones
  - foster **cooperation** among tools

http://www.mrtc.mdh.se/projects/WTC/

Kinds of problems

- Flow analysis and expression
  - loop bound?
  - path is infeasible?
  - input annotations?
- Low-level analysis and WCET computation
  - WCET of a function?
  - number of cache misses on the worst-case path?

Benchmarks

- debie1
  - software of a satellite instrument for measuring impacts of small space debris or micro-meteoroids
  - now publicly available (thanks to Niklas Hoslti)
  - ported to Java by B. Huber/W. Puffitsch
- papabench
  - open-source software of an UAV
  - two communicating programs: autopilot and fbw
  - ported to Java by M. Malohlava
- heli
  - software (student projet) for a helicopter model

Benchmarks (2)

- tcas
  - traffic collision avoidance software
  - used by the software verification community
    - safety properties/requirements
- matmul
  - matrix multiplication
  - intended to exercise cache analysis
- coop
  - synthetic benchmark designed to foster tool cooperation

Benchmarks (3)

benchmark	loop bounds	infeasible paths	annotations	cache analysis	WCET
debie1	15		22		22
papabench	4	11	9		14
heli	1	1	3	2	3
tcas			3		5
matmul				3	2
coop	2	2	1		1

Targets (processors)

- Basic processor
  - 1 instruction = 1 cycle
- ARM7
  - 3-stage pipeline, 10-cycle memory latency
- ARM9
  - 5-stage pipeline, instruction and data caches, 10-cycle memory latency
- MPC5554
  - 4-way unified cache
  - partitioned cache: 1 way for instructions, 1 way for data
- List is open!

Participants so far

- T-CREST (TU Denmark & TU Vienna)
  - M. Schoeberl, A. Jordan, S. Hepp, B. Huber, W. Puffitsch
- gaZ RT toolset (U. of Zaragosa)
  - J. Segarra
- TUBound (TU Vienna)
  - J. Zwirchmayr
- oRange (U. of Toulouse)
  - A. Bonenfant, J. Zwirchmayr
- OTAWA (U. of Toulouse)
  - H. Cassé, C. Rochange

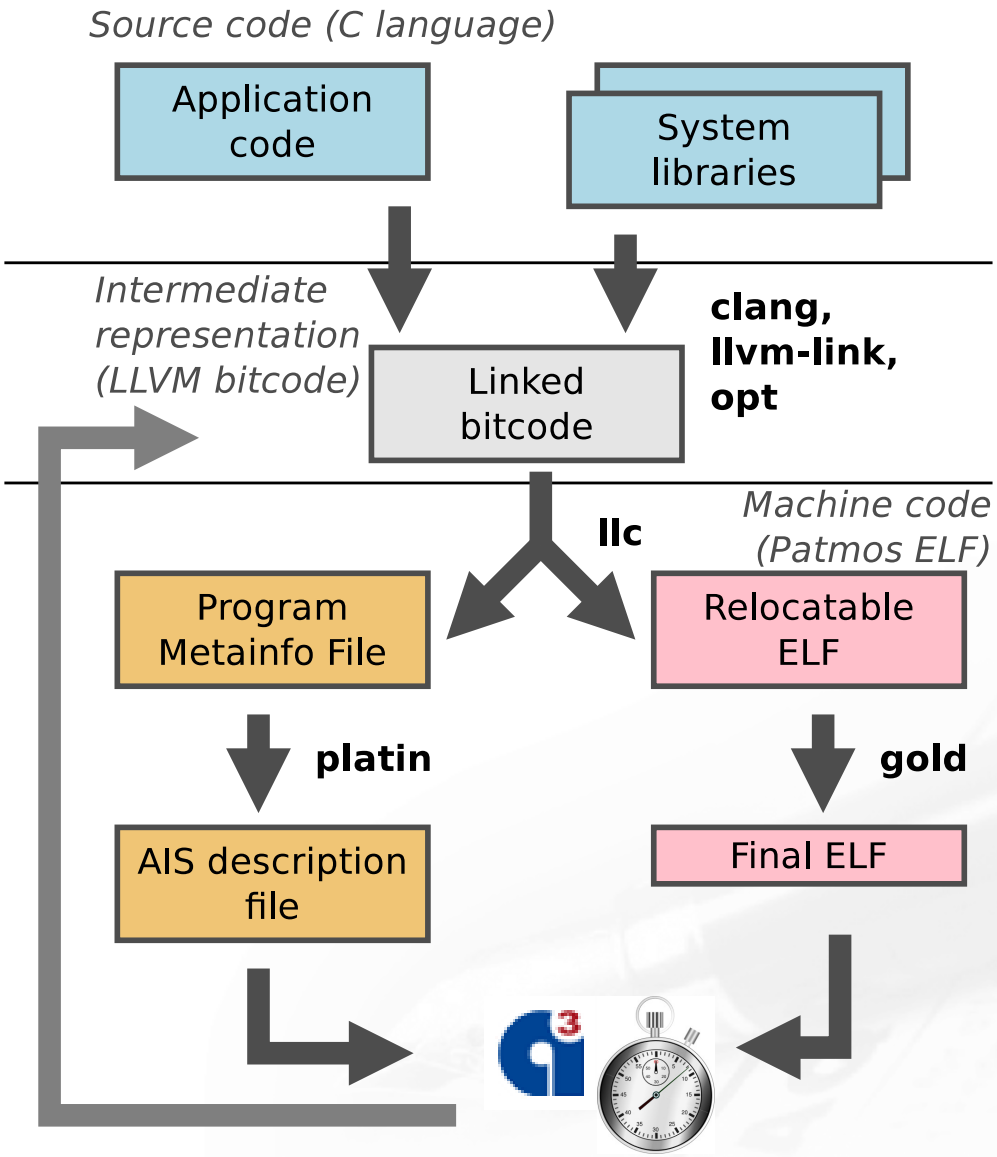
T-CREST

- Time-predictable Multi-Core Architecture for Embedded Systems
- Processor Patmos
- Network-on-chip
- Memory controller and arbitration tree
- WCET optimizing compiler
- WCET analysis with platin/aiT

Patmos

- Time-predictable processor
- In-order RISC
- Static dual-issue for some performance
- Special caches and on-chip memory
  - Method cache
  - Stack cache
  - Data cache + bypass

T-CREST Compiler – WCET Toolflow



T-CREST and WTC

- Demonstrate integration compiler and static analysis
- Disclaimers
  - Platin (especially its flowfact handling) is under development
  - Our numbers (results) can change from every change in our toolchain
- WCET toolchain available open source (except for aiT)
- Our modifications of the WTC benchmarks
- Our test drivers are available
  - [1] https://github.com/t-crest/

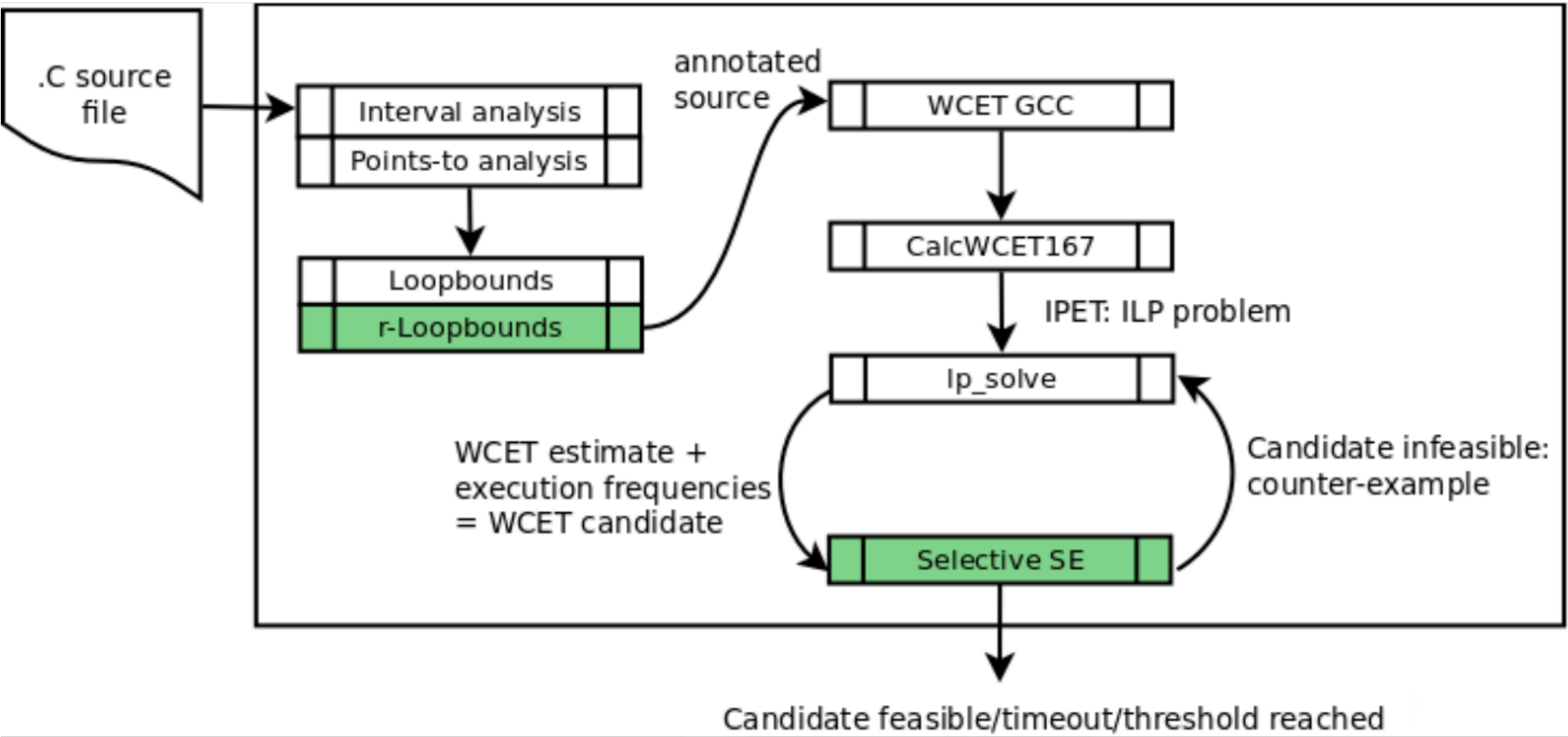
Status

- Focus on new benchmarks: Heli, TCAS, Coop
- Results of compilation and analysis need to be validated (this takes time)
- In our case, programs require indeterminism to be interesting
  - prevent compiler optimizations or trivial analysis)
- We made some modifications (TCAS still needs more)
  - shared our feedback with the WTC organizers

Early Conclusions

- Well-described set of WCET analysis problems
  - New benchmark programs (yay!)
  - Still need few modifications though
- What we learned about our own tools so far
  - Platin can't handle certain CFGs
  - Platin lacks output of some useful analysis information
  - We got ideas on extending Platin's capabilities

TUBound


.C source file
Interval analysis
Points-to analysis
Loopbounds
r-Loopbounds
annotated source
WCET GCC
CalcWCET167
IPET: ILP problem
lp_solve
WCET estimate + execution frequencies = WCET candidate
Candidate infeasible: counter-example
Selective SE
Candidate feasible/timeout/threshold reached

oRange

- Static analysis tool of C programs based on abstract interpretation
  - input:
    - C file(s)
    - some domain values
    - branch choices
    - partial analysis
  - output
    - loop bounds
    - some infeasible paths (related to loop increment variables)
    - critical unbalanced conditionals
    - compatible with OTAWA
    - partial analysis

OTAWA

hardware description (XML)

ISA support (plugins)

PowerPC, Sparc, ARM, M68HC, TriCore

built-in or user-developed analyses

- CFG virtualization (to analyze functions in their call context)
- loop analysis (e.g. dominance)
- instruction and data caches (based on abstract interpretation techniques)
  - ▷ categorisation of instruction fetches and data accesses (*always hit, always miss, loop persistent, unpredictable*)
- pipeline behaviour analysis
  - ▷ execution times of basic blocks

architecture abstraction

program (.elf)

program representation

instructions, basic blocks, CFG

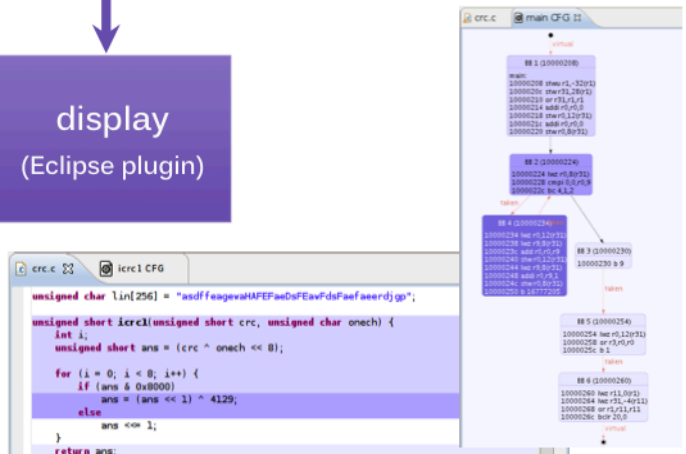
annotations

can be hooked to any object to express built-in or user-defined properties (e.g. WCET of a basic block, loop header, etc.)

flow facts (e.g. loop bounds *)

WCET computation (IPET)

display (Eclipse plugin)



* Our oRange tool determines loop bounds from source code analysis

State of progress (oRange+OTAWA)

- Flow analysis/annotation
  - all problems have been tackled
    - some of them could not be solved
  - exploitation of 'infeasible path' annotations should be improved

- Low-level analysis/WCET computation
  - preliminary results for basic processor, ARM7, ARM9
  - next step: MPC5554 target

Plans for next weeks/months

- New participants are still welcome
  - submitting results is permanently possible
- Additional problems & benchmarks
  - cache locking, flow analysis on binary code
  - lift controlling software
- A report paper will be written within 2–3 months with participants
  - discussion on tools, strong and weak points
  - discussion on benchmarks
  - unsolved problems