Static Analysis for Resource Usage and the COSTA System for Java Bytecode

Germán Puebla[2] and Lars-Åke Fredlund[2]

in collaboration with

Elvira Albert[1], Puri Arenas[1], Samir Genaim[2], Diana Ramírez[2], and Damiano Zanardini[2]

[1] DSIC, Complutense University of Madrid, Spain

[2] CLIP, Technical University of Madrid, Spain

November 2008

Dimensions of Components

As mentioned in previous lectures, when specifying components we need to address the following dimensions:

- Input/output types
- Functional behaviour
- Concurrent behaviour
- Timing behaviour
- Resource usage
- Security

In this lecture we will focus on

- Timing behaviour
- Resource usage

Approaches to Formal Reasoning About Components

As we will see later in the course, there are several approaches to formally reasoning about components. They include:

- Testing
- Model checking
- Formal Verification
- Static Analysis

In this lecture we focus on Static analysis, whose main features are:

- Fully automatic. Termination guaranteed: static approach
- Results cover all possible executions: safe approximations
- Possibly imprecise due to abstraction

Kinds of Cost Analysis

- Different kinds of cost can be considered:
  - worst case $\longrightarrow$ upper bound
  - average case $\longrightarrow$ requires probabilistic study
  - best case $\longrightarrow$ lower bound
- Two classes of upper bounds can be considered:
  - non-asymptotic (or concrete, or micro-analysis)
  - asymptotic (or macro-analysis)
- Analysis results can be
  - platform-independent
  - platform-dependent $\longrightarrow$ WCET

State of the Art in Automatic Cost Analysis

- Work on automatic cost analysis dates back to 1975, with the seminal work of Wegbreit.
- His system, metric was able to compute:
  - interesting results, but for
  - restricted class of functional programs
- Also, the seminal work on abstract interpretation [Cousot & Cousot'77] mentions performance analysis as an application.
- Since then, a number of analyses and systems have been built which extend the capabilities of cost analysis:
  - functional programs [Le Metayer'88, Rosendahl'89, Wadler'88, Sands'95, Benzinger'04]
  - logic programs [Debray and Lin'93, Navas et al'07]
  - imperative programs [Adachi et al'79, Albert et al'07]

A Classical Approach to Cost Analysis

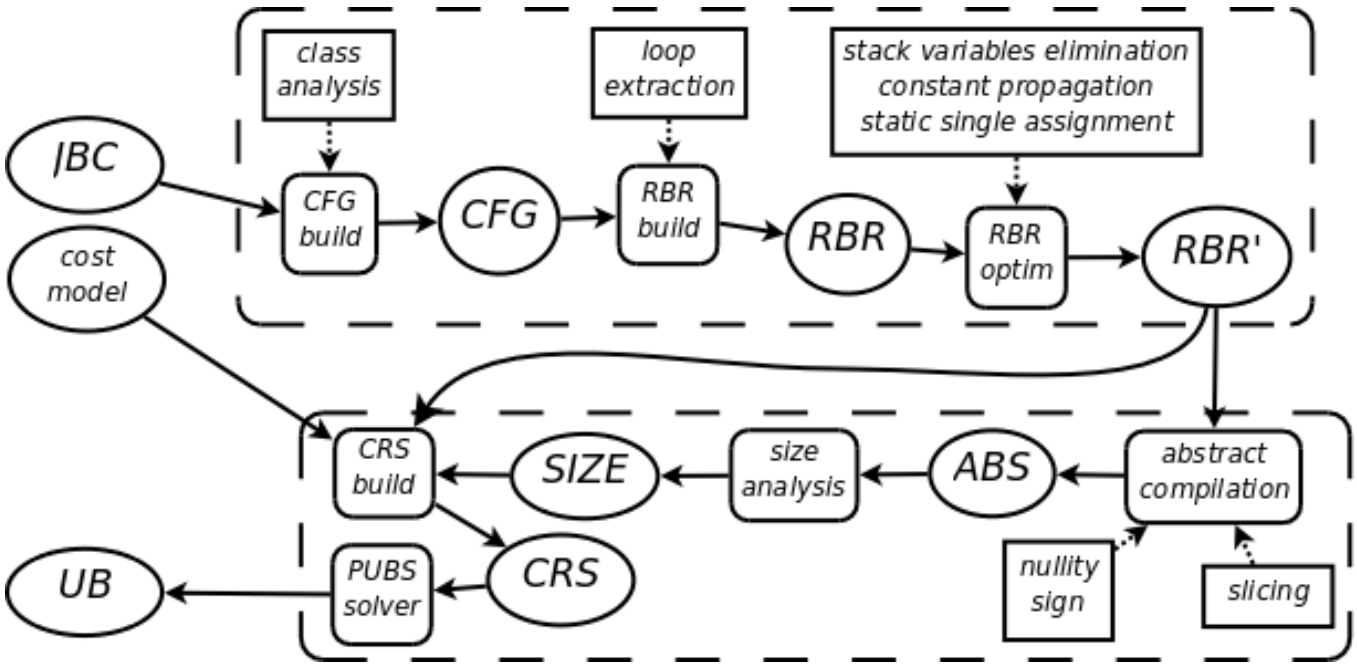
A classical approach [Wegbreit'75] to cost analysis consists of:

1. expressing the cost of a program part in terms of other program parts, thus obtaining *recurrence relations*
2. solving the relations by obtaining a *closed-form* for the cost in terms of the input arguments

The current situation is that

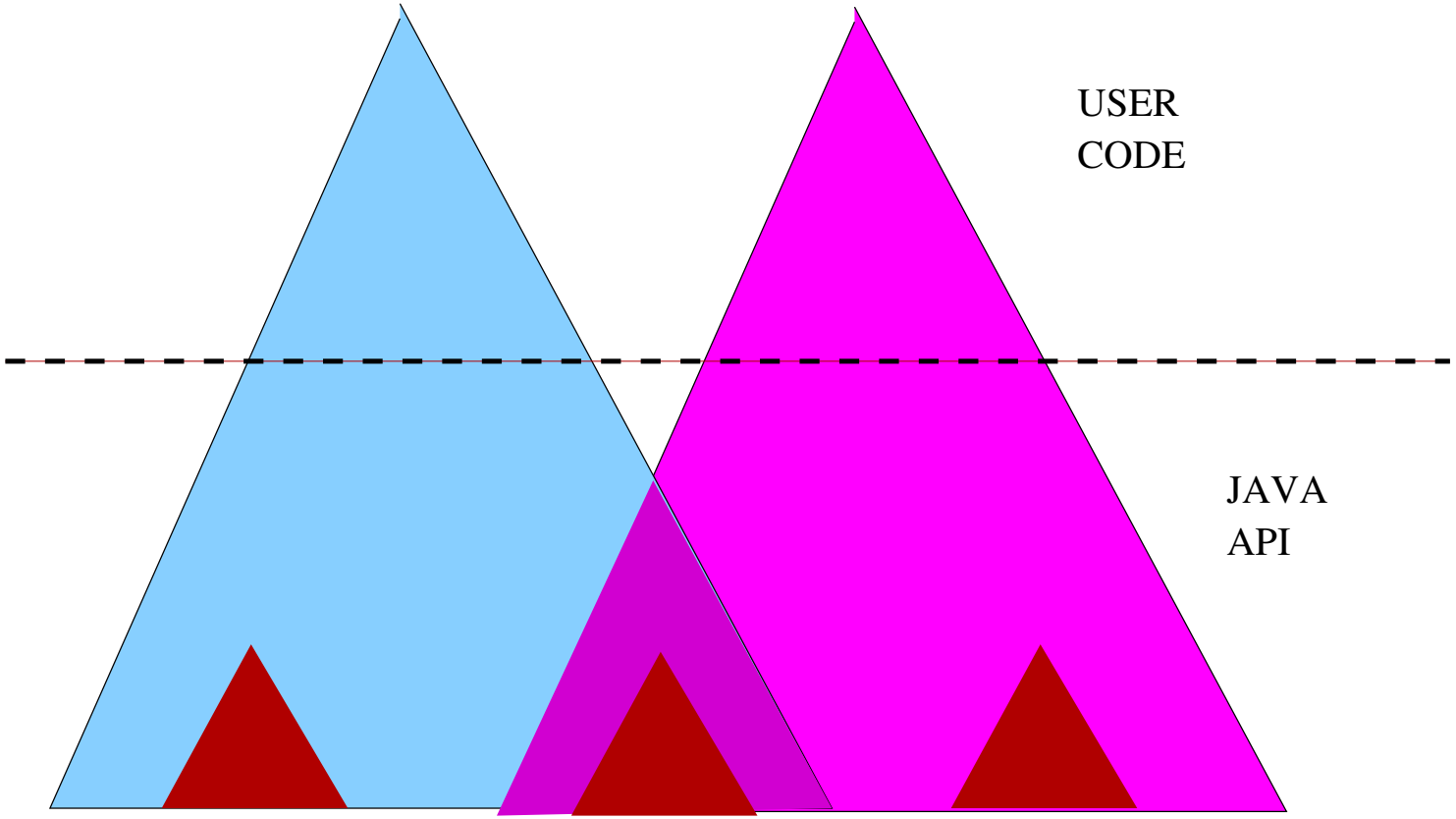
- Most work has concentrated on the first phase
- The difficulties of the second phase have been overseen
- Practical usage of cost analysis requires both!
- In COSTA we address both phases.

The COSTA System

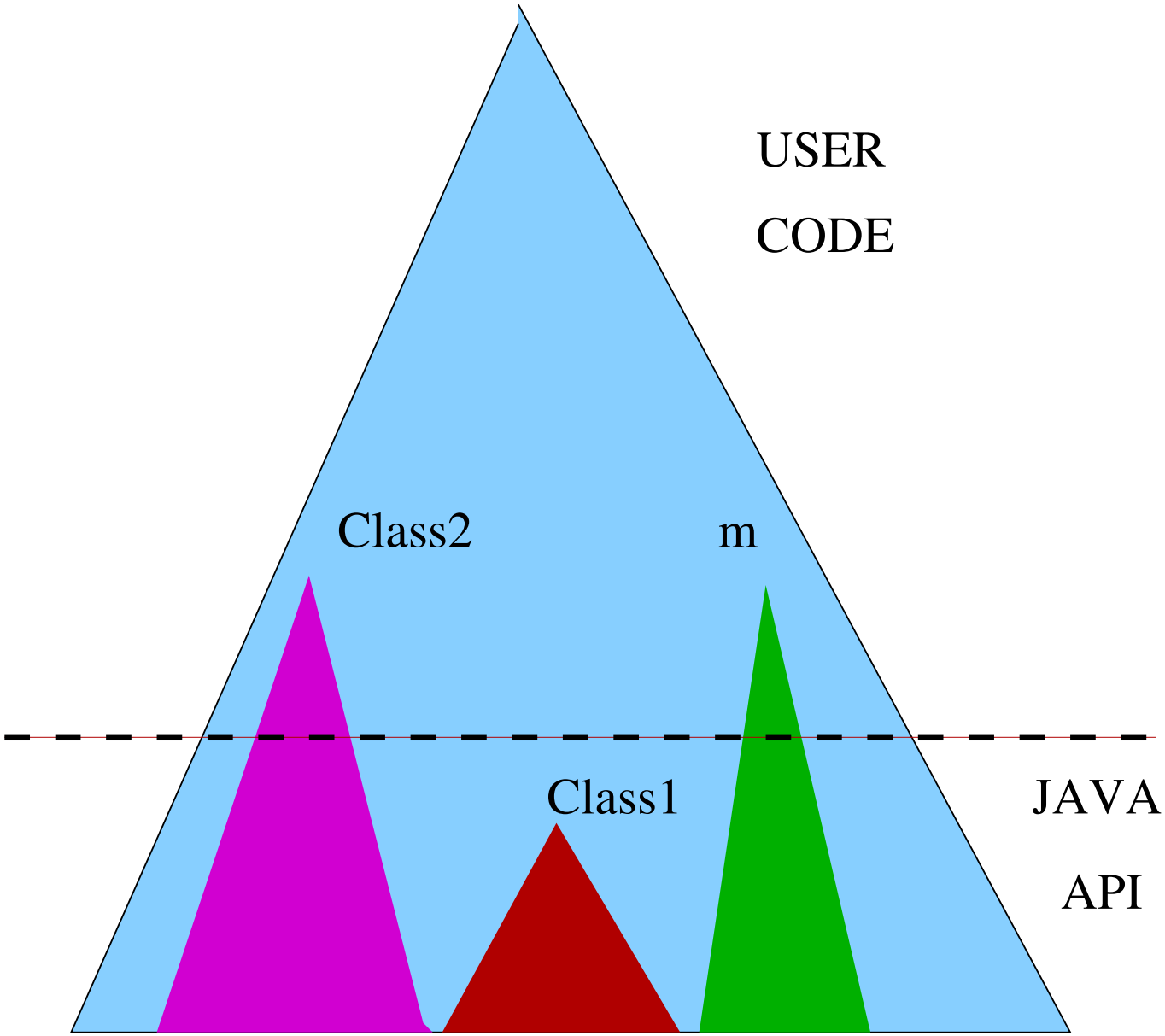


- COSTA: COSt and Termination analyzer for Java Bytecode
- INPUT: bytecode + cost model
- OUPUT: upper bound on resource consumption + termination info

Application Extraction from a `main` method



Application Extraction from an Intermediate Method



Conclusions and Future Work

- The first cost analysis framework for Java bytecode, including
  - unstructured control flow, operand stack
  - exceptions, objects, and virtual method invokation
  - parametric w.r.t. the cost model
- It produces cost expressions for methods in terms of the *size* of their input arguments
- Still a prototype, but quite promising results already!
- COSTA: **COS**T AND **T**ERMINATION **A**NALYZER
  - can be tried out at `http://pargo.ls.fi.upm.es/costa`
  - it will soon be available at the MOBIUS tools site `https://mobius.ucd.ie/trac`
- PUBS: **P**RACTICAL **U**PPER **B**OUNDS **S**OLVER
  - The recurrence relation solver, language independent can be tried out at `http://pargo.ls.fi.upm.es/pubs`
- For more information on the upcoming tool release, please contact the authors.