Essential Ingredients for a WCET Annotation Language

Timing Analysis Cluster Meeting

Munich, March 13, 2007

Raimund Kirner

TU Vienna

Joint work within ARTIST2 of the Compilers Activity and the Timing Analysis Activity

Raimund Kirner, Albrecht Kadlec, Adrian Prantl, Markus Schordan, Jens Knoop, Peter Puschner

Overview

- The WCET Annotation Language Challenge
- Fundamentals
- First steps towards a common WCET annotation language
- Attributes of timing information
- List of timing information
- Future work

The WCET Annotation Language Challenge

- Announced in July 2007
- Motivation:

  Mastering the WCET Annotation Language Challenge is essential for consolidating and advancing the state-of-the-art:

  Precision and performance of WCET analysis depends on expressiveness and usability of the annotation language
- Goal:

  Define a common WCET annotation language to enable the annotation of benchmarks in a tool-independent way.

Fundamentals

- Using the name of “timing annotations” rather than “WCET annotations”
- The annotation language does not need to be as expressive as the programming language
  - only properties of the program behavior have to be described, there is no need to describe the full program semantics
  - avoiding unnecessary expressiveness in the annotation language keeps the WCET analysis efficient.
- Distinction between timing information and timing annotations
- Flow information is subset of timing information

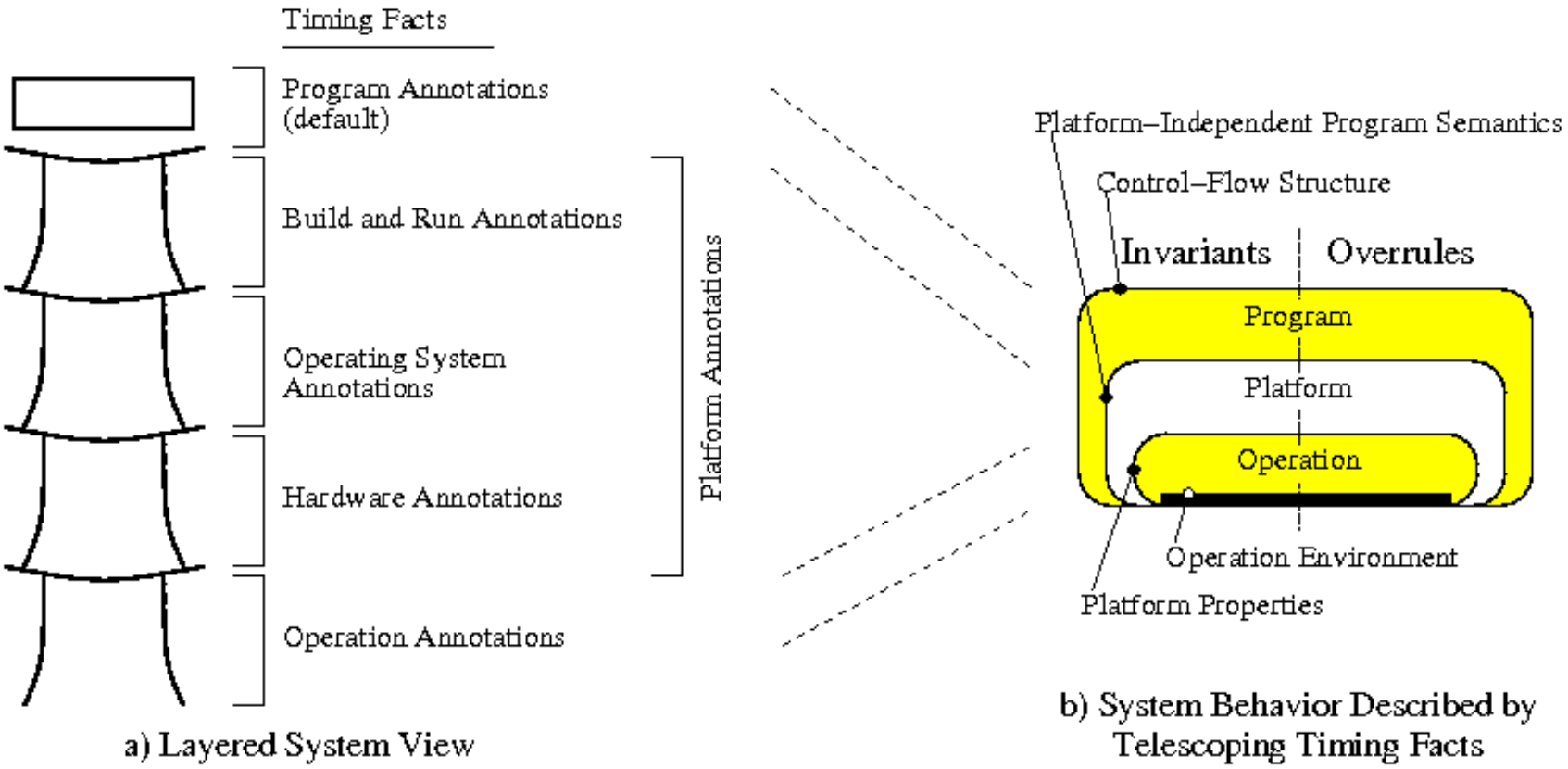
First Steps Towards a Common Annot. Lang.

- Analysis of existing tools and papers to extract timing annotation constructs
- Description of existing annotation constructs in a language-independent way
- Identification of additional constructs
  (e.g., invariants/overrules, annotation layers, selective use by grouping)
- Timing information that is not connected with the program code is left out of the annotation language
- Summarization of the results in a technical report to collect feedback

Attributes of Timing Information

- Annotation class: invariant vs. overrule
  SB_F…feasible system behavior
  invariant: $SB_F \subseteq SB(I_{inv})$
  overrule: $\neg(SB_F \subseteq SB(I_{ovr}))$
  overrules can be used to describe application modes
- Annotation layer:
  program, platform, operation
- Annotation groups:
  - symbolic name with a textual description
    (no predefined semantics)
  - groups may be nested
  - use: selective use of timing information (e.g., multiple platforms, different sets of overrules)

Layered Timing Annotations



different annotation layers

invariants vs. overrules

High-level Annotations

- Loop bounds
- Recursion bounds
- Linear flow constraints
- Variable value restrictions
- Summaries of external functions
  (e.g., side effects, value ranges of results)
  - used to describe an external function for which no code is available

Addressable Units

- Control-flow addressable units
  - basic blocks
  - control-flow edges
  - subgraphs
- Loop contexts
- Call contexts
- Control-flow paths

Control-flow Information

- Specification of unreachable code
- Specification of predicate evaluation
- Control-flow reconstruction

Hardware-specific Low-level Annotations

- Specification of clock rate
- Specification of memory map and memory accesses
- Absolute time bounds

Future Work

- Discussion within the ARTIST2 Timing Analysis Activity to get a common view on the requirements of a timing annotation language.
- Setting up a homepage to collect and publish different proposals towards the common timing annotation language.
- Local meetings with WCET tool vendors and research groups to discuss the instantiation of concrete timing annotation languages.