Predictable Timing of Cyber-Physical Systems
Future Research Challenges

DREAMS Seminar, EECS, UC Berkeley
January 17, 2012

David Broman
broman@eecs.berkeley.edu
EECS Department
UC Berkeley, USA

Department of Computer and Information Science
Linköping University, Sweden

Agenda

Part I
Semantic gap regarding time

Part II
Bridging the gap –
the PRETIL project

Part III
Utilizing a bridged gap –
virtual optimization of CPS
Part I
Semantic gap regarding time

Modeling Cyber-Physical Systems

Physical system (the plant)  Embedded systems (computation)

Part I
Semantic gap regarding time
Part II
Reducing the gap – the PRETIL project
Part III
Utilizing a bridged gap – virtual optimization of CPS

Equation-based model

Abstraction “physical modeling”
Domain-Specific Language (DSL)

- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

Models and Objects

- Object in e.g., Java, C++: object = data + methods
- Objects in EOO languages: object = data + equations

Equation-Based Object-Oriented (EOO) Languages

- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

EOO model (textual)
EOO model (graphical)

- Object in e.g., Java, C++: object = data + methods
- Objects in EOO languages: object = data + equations
Equation-Based Object-Oriented (EOO) Languages

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Domain-Specific Language (DSL)

- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

Equation-Based Object-Oriented (EOO)

Models and Objects

- Object in e.g., Java, C++:
  object = data + methods
- Objects in EOO languages:
  object = data + equations

Acausality

- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

Equation-Based Object-Oriented (EOO) Languages

- Primarily domain:
  Modeling of physical systems
- Multiple physical domains:
  e.g., mechanical, electrical, hydraulic

Models and Objects

- Object in e.g., Java, C++:
  object = data + methods
- Objects in EOO languages:
  object = data + equations

Acausality

- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Equation-Based Object-Oriented (EOO) Languages

Domain-Specific Language (DSL)

- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

Equation-Based Object-Oriented (EOO)

Models and Objects

- Object in e.g., Java, C++:
  object = data + methods
- Objects in EOO languages:
  object = data + equations

Acausality

- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

Equation-Based Object-Oriented (EOO) Languages

- Primarily domain:
  Modeling of physical systems
- Multiple physical domains:
  e.g., mechanical, electrical, hydraulic

Models and Objects

- Object in e.g., Java, C++:
  object = data + methods
- Objects in EOO languages:
  object = data + equations

Acausality

- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Equation-Based Object-Oriented (EOO) Languages
**Part I**
Semantic gap regarding time

**Part II**
Reducing the gap – the PRETIL project

**Part III**
Utilizing a bridged gap – virtual optimization of CPS

---

**Equation-Based Object-Oriented (EOO) Languages**

**Domain-Specific Language (DSL)**
- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

**Models and Objects**
- Object in e.g., Java, C++: object = data + methods
- Objects in EOO languages: object = data + equations

**Equation-Based Object-Oriented (EOO)**

**Acausality**
- At the equation-level: $u = R \cdot i$
- At the object connection level

**Equation-Based Object-Oriented (EOO) Languages**

- **Modelica**
- **VHDL-AMS**
- **gPROMS**
- **MKL**
- **(SPICE)**
Modeling Cyber-Physical Systems

Part I
Semantic gap regarding time

Part II
Reducing the gap –
the PRETIL project

Part III
Utilizing a bridged gap –
virtual optimization of CPS

Modeling the Systems for Computing and Networking

Ptolemy II
Heterogenous modeling environment supporting many different
models of computation (MoC). For example, synchronous dataflow
(SDF), discrete-event (DE), process networks (PN), etc.

PTIDES
Currently implementation in Ptolemy. Modeling of event-based real-
time distributed systems. Based on DE semantics.

Synchronous reactive languages
For example, Lustre, Signal and Esterel

Next versions of Modelica
New semantics for synchronouse discrete semantics
(for improved code generation).

Simulink

And all other languages/environments not listed here…
Model

Equation-based model

Different models of computation

System

Physical system (the plant)

Embedded systems (computation)

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Simulation the CPS

Model

Equation-based model

Different models of computation

System

Physical system (the plant)

Embedded systems (computation)

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Simulation the CPS

Model

Equation-based model

Different models of computation

System

Physical system (the plant)

Embedded systems (computation)

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS
Predictable Timing of Cyber-Physical Systems
meaning that the continuous-time timing behavior for

SIL simulation ≠ HIL simulation ≠ Real-time system execution

Note that predictability is a continuum. The “cyber” can be made deterministic, but the physics cannot.
Part II
Bridging the gap –
the PRETIL project

Precision-Timed Intermediate Language (PRETIL)
High-level requirements

Modelica / MKL
Ptolemy II / PTIDES
Other MoC and tools

Make code generation from source language to PRETIL simple (e.g., via suitable API)
Support multiple modeling (source) languages
Expose language constructs for (physical) execution time

PRETIL

PRETIL compiler

Hide (abstract away) architecture dependent details (e.g., scratchpad)
Formal semantics – reason about correctness of execution time
Enable comparison of platforms

Part I
Semantic gap regarding time
Part II
Reducing the gap – the PRETIL project
Part III
Utilizing a bridged gap – virtual optimization of CPS
Execution time – a correctness factor

Part I
Semantic gap regarding time

Part II
Reducing the gap – the PRETIL project

Part III
Utilizing a bridged gap – virtual optimization of CPS

Sketch - primitives for handling time
(pseudo-code, part of research to be performed)

\( F(x_1, \ldots, x_n) \) is a function in the language with \( n \) parameters.

Static Usage of execution time

- Propagating WCET info up the tool chain:
  - For meta-programming (static scheduling)
  - For tool support (e.g., show WCET for specific actors in Ptolemy)

- Propagate time constraint downwards
  
  \[
  \text{constraint } WCET(f) < 10ms
  \]
  
  Execute with padding (exact time)
  
  \[
  \text{execute } f(3,2) \text{ during } 10ms
  \]

Dynamic usage of execution time

- Execute with padding without guarantees
  
  \[
  \text{execute } f(3,2) \text{ during } 10ms \text{ else } \]

- Use WCET/BCET info dynamically in the model/program.
  
  \[
  \text{if } WCET(f) > 10ms \text{ then } \ldots \text{ else } \]

- WCET of parameterized functions in runtime using parametric WCET analysis (Lisper, 2003)
  
  \[
  \text{if } WCET(f(x_3 = v)) > 10ms \text{ then } \ldots \text{ else } \]
Proposed Infrastructure Overview

Part I: Modeling language front end

Part I: Semantic gap regarding time

Part II: PRETIL front end

Part II: Reducing the gap – the PRETIL project

Part III: PRETIL backend

Part III: Utilizing a bridged gap – virtual optimization of CPS

Part IV: Runtime environment

Part IV: Runtime environment

Part I – Modeling language front end

Research challenge 1: To design (or extend) an intermediate language that hides architecture details and exposes language constructs for programming with (physical) execution time.
Proposed Infrastructure Overview

Part I: Modeling language front end

Part II: PRETIL front end

Part III: PRETIL backend

Part IV: Runtime environment

Research challenge 2: To statically guarantee that timing constraints defined for high-level models hold during run-time.

Formally verified compilers (Leroy, 2009)

Translation Validation Infrastructure (Necula, 2000)
Proposed Infrastructure Overview

Part I: Semantic gap regarding time

Part II: Reducing the gap – the PRETIL project

Part III: Utilizing a bridged gap – virtual optimization of CPS

Part III – PRETIL Back end

Research challenge 3: To optimize allocation of bounded memory resources so that both memory constraints and timing constraints hold simultaneously.
Proposed Infrastructure Overview

Part I: Modeling language front end
- Static analysis
- Optimization and code generation

Part II: PRETIL front end
- Static analysis
- Code generation
- Optimization

Part III: PRETIL backend
- Instruction selection
- Register allocation
- Code generation

Part IV: Runtime environment
- Timing and memory constraints proof validation
- Timing analysis and certification

Research challenge 4: To guarantee safe execution concerning timing of a deployed binary of machine code, without trusting the correctness of the compiler, e.g., by executing a lightweight safety proof before executing the binary.

Proof-carrying code (Necula, 1997)
Part III
Utilizing a bridged gap – virtual optimization of CPS

Simulation with Predictable Timing

The PRETIL project aims at adding one piece of the puzzle to getting predictable timing of CPS.
Design optimization problems

- Parameter optimization of physical objects (e.g., thickness of shafts)
- Architecture parameters, e.g., minimize clock frequency to lower energy consumptions.

Predictable timing with correct timing constraints are essential to performing the optimization on a global CPS model

Hard problems. One approach is to combine CPS simulation with local search heuristics (e.g., tabu search or simulated annealing).

Conclusions and Summary

New project in the Ptolemy group (starting Jan 2012).

Overall challenge
To establish a new formal foundation of timing predictability for the semantics of correct translation/compilation from high-level CPS modeling languages down to machine code for PRET machines.