Heterogeneous Programming and Modeling of Cyber-Physical Systems

Smart Programming Day, SICS Software Week
November 29, 2016

David Broman
Associate Professor, KTH Royal Institute of Technology

Collaborators: Jeremey Siek (Modelyze) and Saranya Natarajan (Timed C)

Examples of application areas

- Automotive (systems of systems)
- Industrial Automation
- Aircraft (traditional or autonomous)
- Satellites
- Medical Equipment

Cyber-Physical Systems (CPS)
Heterogeneous Model-Based CPS Design

Research Challenge #1:
• Main challenge concern heterogeneous mixing of formalisms

Research Challenge #2:
• Main challenges concern
  • Correct handling of time
  • Target heterogeneous hardware

A main challenge
• heterogeneous mixing of formalisms

Physical system (the plant)
Cyber system: Computation (embedded) + Networking

Model
Equation-based DSLs
Virtual Testing: Time-aware simulation

Physical prototyping
Modeling

System
Sensors
Actuators

Physical system (the plant)
Cyber system: Computation (embedded) + Networking

Model
Equation-based DSLs
Virtual Testing: Hardware in-the-loop simulation

Physical prototyping
Modeling

System
Sensors
Actuators

Physical system (the plant)
Cyber system: Computation (embedded) + Networking

David Broman, dbro@kth.se


**Equations and initial values are defined declaratively, just as the mathematical equations**

```python
def Pendulum(m: Real, l: Real, angle: Real) = {
def x, y, T: Real;
init x (l*sin(angle));
init y (-l*cos(angle));
T*x/l = m*x'';
-T*y/l - m*g = m*y'';
x^2 + y^2. = l^2.;
}
```

Differential-Algebraic Equations (DAEs).

\[ -T \cdot \frac{x}{l} = m \ddot{x} \quad x(0) = l \sin(\theta_s) \]
\[ -T \cdot \frac{y}{l} - mg = m \ddot{y} \quad y(0) = -l \cos(\theta_s) \]

\[ x^2 + y^2 = l^2 \]
Declarative Mathematical Model

Which parts are part of the host language (Modelyze)?

Unknowns are internally represented as typed symbols.

Syntax and semantics for differential equations are embedded into the host language Modelyze.

Embedding and Execution Process

Static Semantics

- Model Libraries
- Model
- DSL Semantics

Dynamic Semantics

- Type Checking
- Symbol Lifting Analysis
- Eval into Symbolic Expr
- Analyze and Transform
- Interpretation, Partial Evaluation and Execution

Deep embedding:
Manipulation of symbolic expressions

Shallow embedding:
Types, functions, etc. of the host language are used directly as part of the DSL.
Embedding and Execution Process

**Deep embedding:** Manipulation of symbolic expressions

**Shallow embedding:** Types, functions, etc. of the host language are used directly as part of the DSL

**Cheap embedding**
The aim of combining the convenience of shallow embedding with the power of deep embedding.

Other names for combining shallow and deep embedding: neritic (Augustsson, 2012) and Yin-Yang in Scala (Jovanovic et al., 2014)
What is our goal?

“Everything should be made as simple as possible, but not simpler”
attributed to Albert Einstein

**Execution time should be as short as possible, but not shorter**

No point in making the execution time shorter, as long as the deadline is met.

**Objective:**
Minimize area, memory, energy.

**Challenge:**
Still guarantee to meet all timing constraints.

---

Programming Model and Time

Timing is not part of the software semantics
The correctness of programs is not related to execution time.

**Traditional Approach**

Programming Model

Timing Dependent on the Hardware Platform

**Our Objective**

Programming Model

Make time an abstraction within the programming model

Enable timing portability, where timing requirements are verified by the compiler.
Compilation and Analysis

Timed and/or synchronous modeling languages (Simulink, Modelica, Ptides, SCADE etc.)

Timed C programs

May act as an intermediate language.

Source-to-source compilation

C Compiler

Distributed real-time high performance platforms (RTOS, POSIX API, Linux, Windows)

Timing Analysis

C Compiler

Distributed embedded microcontrollers (bare metal implementations)

Conclusions

Some key points:

**Modelyze** is an ongoing project for embedding heterogeneous domain-specific modeling languages.

**Timed C** is an ongoing project where we incorporate real-time into low level languages.

For more info, visit our group page:

**Model-based Computing Systems (MCS)**

http://www.kth.se/ict/mcs

Thanks for listening!