Zélus: a synchronous language with ODEs

Timothy Bourke\textsuperscript{1,2}  Marc Pouzet\textsuperscript{2,1}

1. INRIA Paris-Rocquencourt
2. École normale supérieure (DI)

http://www.di.ens.fr/ParkasTeam.html

HSCC 2013, CPS Week, April 8–11, Philadelphia, USA
Hybrid Systems Modelers

Program complex discrete systems and their physical environments in a single language

Many tools exist

- Simulink/Stateflow, LabVIEW, Modelica, Ptolemy, ...

Focus on programming language issues to improve safety

Our proposal

- Build a hybrid modeler on top of a synchronous language
- Recycle existing techniques and tools
- Clarify underlying principles and guide language design/semantics
Typical system

Physical environment

- ODEs with reset
  \[ \text{der } v = (0.7 \div \text{maxf}) \times \text{error} \]
  \[ \text{init } 0.0 \quad \text{reset } \text{hit}(0) \rightarrow v0 \]

- Hierarchical hybrid automata

Discrete controller

- Dataflow equations
- Hierarchical automata
Reuse existing tools and techniques

Synchronous languages (SCADE/Lustre)

- Widely used for critical systems design and implementation
  - mathematically sound semantics
  - certified compilation (DO178C)
- Expressive language for both discrete controllers and mode changes

Off-the-shelf ODEs numeric solvers

- Sundials CVODE (LLNL) among others, treated as black boxes
- Exploit existing techniques and (variable step) solvers

A conservative extension:

Any synchronous program must be compiled, optimized, and executed as per usual
Type systems to separate continuous from discrete

What is a discrete step?
- Reject unreasonable parallel compositions
- Ensure by static typing that discrete changes occur on zero-crossings
- Statically detect causality loops, initialization issues

Simulation engine

\[
\begin{align*}
\sigma' &= d_\sigma(t, y) \\
\text{upz} &= g_\sigma(t, y) \\
\dot{y} &= f_\sigma(t, y)
\end{align*}
\]
Compiler architecture

Built on an existing synchronous compiler

- Source-to-source and traceable transformations
- Resulting program is synchronous and translated to sequential code

code generation

scheduling optimization

last/fby/ → ODEs zero-crossings

variable completion

present/s signals
discrete zero-crossing

periods

normalize let/in

automata
Comparison with existing tools

Simulink/Stateflow (Mathworks)

- Integrated treatment of automata vs two distinct languages
- More rigid separation of discrete and continuous behaviors

Modelica

- Do not handle DAEs
- Our proposal for automata will be integrated into new version 3.4

Ptolemy (E.A. Lee et al., Berkeley)

- A unique computational model: synchronous
- Everything is compiled to sequential code (not interpreted)
Zéus: A Synchronous Language with ODEs

Timothy Bourke    Marc Pouzet
INRIA Team PARKAS, École normale supérieure (Paris, France)
http://www.di.ens.fr/ParkasTeam.html

Programming embedded systems and their environments in the same language

- A Lustre-like language with ODEs
- Dedicated type system to separate discrete time from continuous time behaviors.
- A compiler architecture based on checkable source-to-source transformations.
- Simulate with an off-the-shelf numeric solver.

Hybrid simulation run-time

The Type system

\[
\begin{align*}
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\forall k & : \mathbb{R} \\
\end{align*}
\]

Example system with (hierarchical) Hybrid Automaton

Hybrid Systems: Computation and Control
9-11 April 2013
Philadelphia, USA