Sound and Soundness
Practical Total Functional Data-Flow Programming

Baltasar Trancón y Widemann$^{1,2}$  Markus Lepper$^2$

$^1$ Programming Languages & Compilers
Ilmenau University of Technology

$^2$ semantics GmbH

September 6, 2014
Our Background

- Compiler construction
  - declarative domain-specific
  - multi-paradigm, embedded
  - meta-programming

- Application fields
  - scientific modeling & simulation
  - real-time, embedded & control
  - music

- The SÎG language and system
Our Background

- Compiler construction
  - declarative domain-specific
  - multi-paradigm, embedded
  - meta-programming

- Application fields
  - scientific modeling & simulation
  - real-time, embedded & control
  - music

- The S\(\Sigma\)G language and system
Our Background

- Compiler construction
  - declarative domain-specific
  - multi-paradigm, embedded
  - meta-programming

- Application fields
  - scientific modeling & simulation
  - real-time, embedded & control
  - music

- The SIG language and system
**SIG is...**

**LIKE** Visual data-flow tools
- for domain expert use
- complex component networks

**BUT** Semantically safe & sound
- types
- purely declarative, reproducible
**Sig is...**

**LIKE** Visual data-flow tools
- for domain expert use
- complex component networks

**BUT** Semantically safe & sound
- types
- purely declarative, reproducible
Yet Another Language? (2)

**ΣιG is...**

**LIKE** French synchronous languages
- rigorous simple semantics (automata)
- specific model of time

**BUT** High-level programming
- abstraction & reuse
- symbolic computation (ADTs)
Sig is...

**LIKE** French synchronous languages
- rigorous simple semantics (automata)
- specific model of time

**BUT** High-level programming
- abstraction & reuse
- symbolic computation (ADTs)
Sig is . . .

**LIKE** Functional (reactive) programming
- implied lifting values $\rightarrow$ signals
- purely functional bells & whistles

**BUT** Discrete, and proud
- clocked time $\Rightarrow$ streams
- foundation in ordinary maths
**SIG is...**

**LIKE** Functional (reactive) programming
- implied lifting values \(\rightarrow\) signals
- purely functional bells & whistles

**BUT** Discrete, and proud
- clocked time \(\Rightarrow\) streams
- foundation in ordinary maths
Current Status

- (Long design phase)
- Semantics & core calculus [MSFP’14]
- Prototype execution environment
  - hosted in Java
  - interpreter
  - jit-compiler to JVM bytecode
\[ \delta \cdot (2 - \varepsilon^2) \]

Diagram with nodes and edges.
\[
\begin{align*}
\dot{x} &= -r(x) \\
&\quad \text{for } x \geq 0 \quad g = 0
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= +a(x) \\
&\quad \text{for } x < 1 \quad g = 1
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= 0 \\
&\quad \text{for } x \leq s \quad g = 1
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= -d(x) \\
&\quad \text{for } x \geq 0 \quad g = 0
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= +a(x) \\
&\quad \text{for } x < 1 \quad g = 1
\end{align*}
\]
Future Vision

- configuration meta-programming
- multi-rate systems & slicing
- domain-specific type systems
- analysis tools
- backends for GPU, DSP, FPGA, ...