Reactive Model-based Programming of Embedded Systems

Prof. Brian C. Williams
Dept. of Aeronautics and Astronautics
Space Systems and Artificial Intelligence Labs
MIT

SDP Panel, December 13th, 2001
Observations: VLSI circa 1979

VLSI designers aren’t good at reasoning through complex physical interactions:

Solution:

• Simplifying abstractions
• Design rules
• Design rule verifiers
• Silicon compilers
Observations: Embedded SW, 2001
Programmers of embedded systems:

• Aren’t good at reasoning through complex physical interactions.
  (Mars Polar Lander, test stand and sw monitor failure).

• Aren’t good at anticipating all novel interactions with the environment.
  (Deep Space One, star tracker).

• Rarely have time to add in fault protection layers.
  (Mars Polar Lander and Climate Orbiters).

- Embedded languages should do this for you.
Thesis: Model-based Programming

Embedded programs should:

• include models of the physical plant.
• reason through plant interactions for you.
• reveal their reasoning at compile time for analysis.
• reason on the fly to handle unanticipated circumstances.
• reason on the fly to optimize performance to the situation.

We should fold extensive reasoning into our interpreters and compilers
Reactive Model-based Programming Language, v 1.0

Embedded programs interact with plant sensors and actuators:

• Read sensors
• Set actuators

Model-based programs interact with plant state:

• Read state
• Write state

Programmer must map between state, sensors, and actuators.

Model-based executive maps between state and sensors/actuators.

Requires: Propositional SAT engine in reactive loop
DOD: On To Cooperative Systems

objective
path planned/taken
identified target
possible target
exclusion zones
exploration region
Reactive Model-based Programming Language, v 2.0

• Cooperative Programs
  • Specify team behaviors as concurrent embedded programs.
  • Introduce redundant options with decision theoretic choice.
  • Introduce timing requirements between activities.

• Model-based Executive
  • Plans and schedules options at the scale of seconds.
  • Continuously searches for optimal plans
  • Monitors execution and replans.

Requires: hierarchical planning and scheduling in reactive loop
Reactive Model-based Programming Language, v 3.0

- **Cooperative Programs**
  - include goal destinations and flight dynamics

- **Model-based Executive**
  - plans trajectories and detailed control actions.

\[ x_{ip} - x_{iq} \geq d \]
\[ \text{or} \quad x_{iq} - x_{ip} \geq d \]
\[ y_{ip} - y_{iq} \geq d \]
\[ \text{or} \quad y_{iq} - y_{ip} \geq d \]

Requires: kino-dynamic path planning and mixed integer/linear programming with in the reactive loop
Embedded systems need to anticipate the seemingly unlikely
Reactive Model-based Programming Language, v N.0

- Model-based Programs
  - same as before

- Model-based Executive
  - tracks unlikely system trajectories.
  - extracts statistically significant trends from noise.
  - checks future safety of most likely trajectories.
  - validates plans against likely failures.
  - plans contingencies and prepares for them.

Requires: hybrid mode estimation, model checking, Bayesian inference…within the reactive loop
Programmers of embedded systems:

- Don’t like reasoning through interactions and failure.

- **Embedded languages should do this for you.**

We should fold extensive reasoning into our online interpreters and compilers.