The Timed Abstract State Machine (TASM) Language and Toolset

Martin Ouimet, Guillaume Berteau, Kristina Lundqvist
Embedded Systems Laboratory
Massachusetts Institute of Technology
October 1st, 2006
Introduction

- Hi-Five Framework
  - Specification framework for V & V of Real-Time Embedded Systems
  - Use same language across components and abstraction levels
  - Be “open”
  - Be “usable”
The Timed Abstract State Machine Language (TASM) - I

- Derivative of Abstract State Machines (ASM)

- Extensions to include non-functional behavior alongside functional behavior
  - Time
  - Resource consumption

- "Literate" specification language
The Timed Abstract State Machine Language (TASM) - II

- ASM + Time + Resources
- Parallel and hierarchical composition
- Synchronous and asynchronous systems
- Synchronization model
  - Time
- Communication model
  - Shared variables
TASM – Global View

ENVIRONMENT

ASM 1

ASM 2

ASM 3
Introducing ASMs – States and Update Sets

- **State** = Value of environment variables at a given point in time, a “step”
- **Step** = ASM produces an update set, that is an assignment of one or more values to one or more global variable
- **Run** = Sequence of update sets
- **State Evolution** = Application of update set at each step of the run. $S_i = S_{i-1} @ U_i$
- **Example:**
  - Run: $U_1, U_2, U_3, U_4, \ldots$
  - States: $S_0, S_1, S_2, \ldots$
Introducing TASMs – State and Update Sets + Time + Resources

- State = Value of environment variables at a given point in time, a “timestamp” on the global time axis
- State Evolution = Application of update set at specific timestamps of the run. \( T_{S_i} = T_{S_{i-1}} \oplus T_{U_i} \)
- Example:
  - Run: \( T_{U_1}, T_{U_2}, T_{U_3}, T_{U_4}, \ldots \)
  - States: \( T_{S_0}, T_{S_1}, T_{S_2}, \ldots \)
  - \((0, S_0), (t_1, S_1), (t_2, S_2), \ldots\)
ASM - Example

- **Initial environment**
  - (light1, On), (light2, Off)

- **Synchronous**
  - (On, Off), (Off, On), (On, Off), etc.

- **Asynchronous**
  - (On, Off), (Off, Off), (On, Off), (Off, Off), etc.
TASM - Example

- \( t = 0 \)
  - (On, Off)
  - memory = 40
- \( \ldots \)
- \( t = 1 \)
  - (On, On)
  - memory = 40
- \( \ldots \)
- \( t = 2 \)
  - (Off, On)
  - memory = 50
- \( \ldots \)
A More Substantial Example – Electronic Throttle Controller

- Drive-by-wire system to regulate throttle angle and fuel injection for fuel-efficiency
- Relies heavily on modes to compute commanded throttle voltage
- Multiple combination of modes – rev-limiting, traction-limiting, cruise-control, human-input
- From Paul G. Griffiths (UC Berkeley)
A More Substantial Example – Electronic Throttle Controller

Diagram:
- Start-up → Human Control → Inactive → Shut-down
- XOR from Start-up to Inactive
- AND from Inactive to Inactive
- XOR from Inactive to Cruise
- XOR from Inactive to Rev Limiting
- XOR from Cruise to Traction Control
- XOR from Rev Limiting to Traction Control

Driving Modes:
- AND from Human Control to Inactive
- XOR from Inactive to Cruise

Limiting Modes:
- XOR from Inactive to Rev Limiting
- XOR from Inactive to Traction Control

The TASM Toolset

- Tool to:
  - Read and write specifications in the TASM language
  - Simulator to execute specifications
  - Verify consistency of rule sets for a given machine
  - Verify completeness of rule sets for a given machine
A More Substantial Example – Electronic Throttle Controller
A More Substantial Example – Electronic Throttle Controller
TASM Toolset Demo

- Simple Example
- ETC
Research Roadmap

- Create more examples to exercise language
- Map to UPPAAL to verify best-case and worst-case behavior (time, resource)
- Use conversion to CNF to generate test-cases
Summary

● Conclusion
  ● TASM language – executable specification language
  ● Unified language – function + time + resources
  ● Toolset for analysis and simulation

● For more information
  ● tasm@mit.edu
  ● http://esl.mit.edu/tasm
References

- **ASM**

- **ETC**