MoBIES kickoff meeting; 27-29 June 2000, Monterey CA

Formal Composition Technology For Time-Triggered Systems

John Rushby,

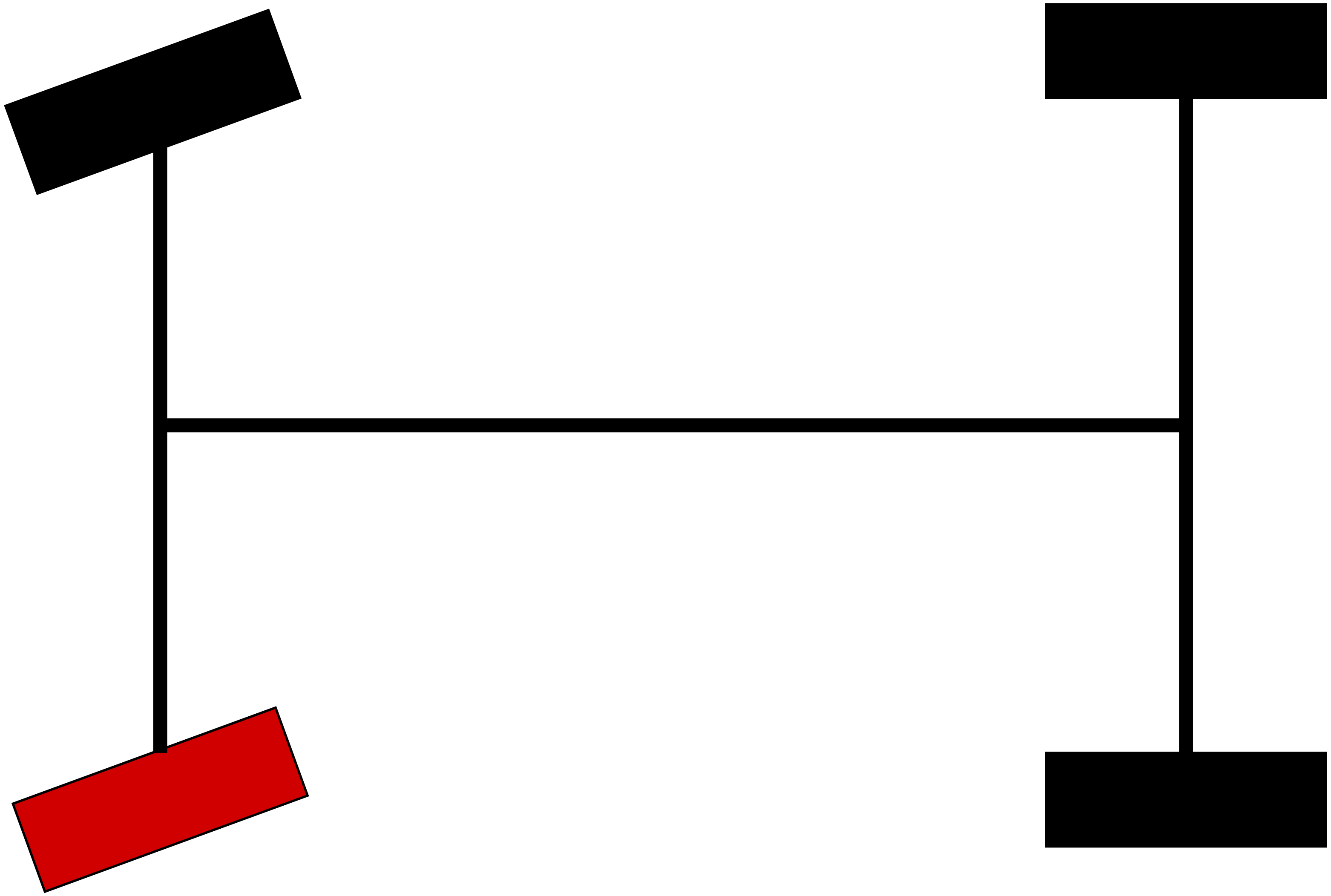
Computer Science Laboratory
SRI International
Menlo Park CA USA

Goals

- Make it possible to build and extend embedded systems for critical applications
- Easier and cheaper than before
- With higher assurance than before
- By using well-understood design frameworks
  - Having assured properties

  In combination (i.e., composition/integration)
- Mechanized formal methods support integration and provide assurance for the composition

Motivating Example



Frameworks

- A design framework provides a conceptual model and a set of mechanisms for putting certain kinds of systems together
- Sacrifice some design freedom in return for operating with a well-understood and well-supported set of ideas
- A framework embodies, but is more than, a set of mechanisms for synchronization, communication, fault-tolerance etc.
- It is a "world view" on how build systems on those mechanisms
- Examples (not embedded): TIB, Horus, COM
- Different frameworks address different concerns: so want to compose several of them

Frameworks For Embedded Systems

- There are several, each aimed at a slightly different domain, or having a different "world view," or operating at a different level of abstraction
- High-level modeling frameworks
  - Matlab/Simulink, Matrix$_x$
- Lower-level design frameworks
  - Data: SCADE/Lustre/Signal/Esterel
    - ⋆ SCADE is used to synthesize all the code on Airbus planes; has DO-178B-approved compiler
  - Control: Statecharts, Stateflow
- Bottom-level frameworks
  - Time-Triggered Architecture (TTA)

Our Project

- Use mechanized formal methods to integrate frameworks at different levels
- E.g., design in Simulink, implement in Lustre and Statecharts, run on TTA
- Formal methods will justify transformations across frameworks
    - And generate glue code
- So that properties established at one level are preserved at the next, and properties of one system are preserved when composed with another
- With particular focus on nonfunctional properties such as timing, fault tolerance, partitioning
    - Because these dominate the development time, assurance costs, and failures in embedded systems

Aside: Mathematics in Engineering

- Engineers build mathematical models of a design, its environment, and requirements
- Use calculation to establish that the design in the context of the environment satisfies the requirements
- Only useful when mechanized (cf. CFD)
- Used in the design loop (exploration, debugging) as well as for verification
- Need to be sure that design is implemented correctly, and that environment and requirements models are "correct"
    - These are separate concerns

Formal Methods

- Same idea
- The applied math of Computer Science is formal logic
- So the models are formal specifications in some logical system: $\mathit{environment} + \mathit{design} \vdash \mathit{requirements}$
- And calculation is mechanized by automated deduction: theorem proving, model checking, etc.
  - Undecidable in general; need human guidance
  - Decidable problems have much greater computational complexity than mechanizing continuous mathematics
- We have state-of-the-art tools for mechanized formal analysis: PVS and SAL (used, e.g., in certifying the shutdown system of the Darlington Nuclear power station)

Lowest Level Framework Will Be TTA

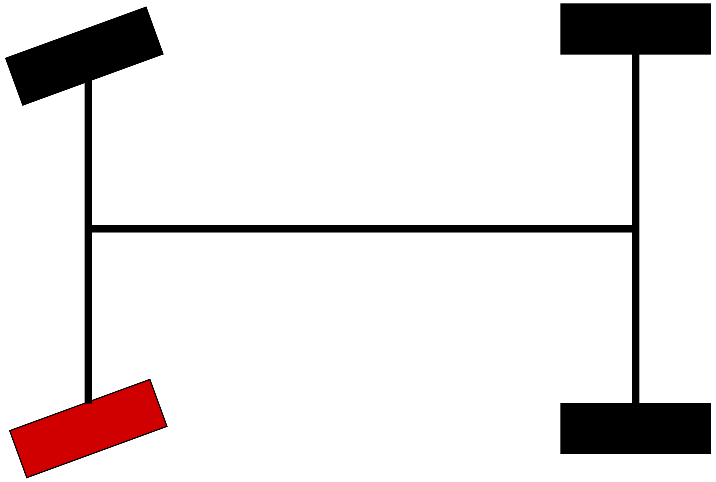
- From TU Vienna and TTTech
- Provides an integrated set of services for safety-critical real-time systems (cf. CAN)
  - Byzantine fault-tolerant clock synchronization
  - TDMA communication (fault tolerant, partitioned)
  - Group membership
  - Transparent application-level fault-tolerance
    - ⋆ Fault-masking based on state-machine replication
    - ⋆ Or fail-stop master/checker etc.
- Becoming widely-accepted in automobile and avionics applications
- Provides COTS-like attributes of low-cost and ubiquity

Formally Verified Mechanisms of TTA

- Clock synchronization (verified in PVS by Pfeifer, Schwier, and Friedrich W. von Henke, DCCA'99)
- Group membership (verified in PVS by Rushby, CAV'00 and Pfeifer Forte/PSTV'00)
- State machine replication and transient-recovery (verified in predecessor of PVS by Rushby, FTRTFT'92)
- Time-triggered implementation of synchronous algorithms (verified in PVS by Rushby, DCCA'97, TSE'99)

Aside: The Need for Group Membership

- Consider a brake-by-wire application
- Separate computers at each wheel adjust braking force according to inputs from brake pedal, accelerometers, steering angle, wheel-spin sensors etc.
- Suppose one of these computers fails



- The others need to redistribute the braking force
- So must have consistent opinion about who has failed

Formal Models for Other Embedded Frameworks

- Lustre (Caspi et al, in PVS, DCCA'99)
- SPLICE (Hooman et al, in PVS, SAC'00)
- Tabular and Statecharts-like specifications (various, in PVS)
- Model validation for computer algebra systems (Martin et al, in PVS, ISSAC'99)
  - Mathworks are interested in these issues for Stateflow

Integration and Composition

- It's not just a question of specifying APIs
- For example, transparent fault-tolerance doesn't have an API in the conventional sense (that's why it's transparent), but it must be used correctly (e.g., different replicas must run in different fault-containment regions)
- It's these assumptions and nonfunctional aspects that need to be captured and managed correctly as different models are integrated and composed
- However, we will start off by specifying APIs and functional properties (Year 1), then move on to timing (Year 2), and nonfunctional properties (Year 3)

Plan

- Framework library: formal characterizations of the models underlying popular frameworks
- Framework mappings and verified preservation or augmentation of critical properties
- Executable models and transformations: specify requirements relative to a framework and then validate relative to its model
- Automated generation of glue code: based on the PVS evaluator and formally specified transformations
- Experiments and examples: to drive and validate the approach