Introduction

- Reactive Systems
  - All have concurrency
  - Includes embedded systems, real-time systems, some software systems

- MoC’s that support concurrency:
  - Comm. sequential processes (CSP)
  - dataflow process networks (DF)
  - discrete events (DE)
  - synchronous/reactive (SR)
Introduction

- FSM's with concurrent MoC's
  - Statecharts
  - Argos - FSM's and SR model
  - Argos and Lustre
  - SDL - process networks with FSM's
  - CFSDM - FSM's and DE
  - hybrid - continuous with FSM's
- All intertwine concurrency and automata semantics
- Limited compositionality

Introduction

- *charts ("star-charts")
  - decouple concurrency model from semantics
  - can hierarchically combine different concurrency models
- Desirable MoC's properties
  - compositional - composite modules used as primitives
  - heterogeneity - composite modules embedded within foreign MoC's
Finite State Machines

- The Basic FSM
  - deterministic
    - at most one enabled transition for each input
  - reactive
    - at least one enabled transition for each input
    - self-transitions assumed

Finite State Machines

- Multiple Inputs and Outputs
  - Pure
    - size of input symbol set and a power of two
      - $\Sigma^2 \cdot 2^M$
    - size of each signal alphabet = 2
      - ex. signal consists of “event” that is present or absent
Finite State Machines

- **Valued**
  - size of a signal alphabet > 2
  - used in FSM's that have arithmetic operations
- **Hierarchical**
  - slave FSM action first, then master FSM action
  - outputs merged
  - valued FSM: no two triggered transitions can have same output signal

Finite State Machines

- **Hiererarchy and Valued FSM**
  - makes notations more compact, not fundamentally more expressive
Mixing FSM’s

- **Termination**
  - To refine a FSM, MoC must have a finite iteration

Mixing FSM with Dataflow

- **Dataflow with FSM**
  - General DF and PN
    - Finite iteration existence is undecidable
  - SDF
    - Decidable if iteration exists
    - Deadlock existence decidable
    - Special case: homogenous SDF
      - 1 input => 1 output for all arcs
• Example:
  • 2 pure FSM’s in a homogenous SDF
  • FSM’s fired sequentially within 1 SDF iteration
  • FSM can fire concurrently across iterations

{D, C, C, C, E}

Two types of firings:
  • Type A firing
    - No transition, only refined subsystems of current state fired
  • Type B firing
    - Transition taken

Allows FSM to stay in same state for top level SDF iteration
SDF in FSM

- FSM refined to SDF
  - SDF graph iteration followed by FSM reaction
  - Easy if SDF is homogenous

- Non-homogenous SDF’s in FSM
  - Type signature
    - FSM that SDF embedded in treated as SDF
  - Complications
    - SDF composition not always possible
    - FSM (with embedded SDF) not always in SDF
    - Type signature can change in different FSM states
Heterochronous Dataflow

- For type signature changes
- Two sets of balance equations needed
- Type signature constant during HDF iteration
- Compromises modularity

Dynamic Dataflow

- FSM in DDF
  - Fire FSM until input tokens consumed
- DDF in FSM
  - Very complicated, not discussed
Discrete Events and FSM

- DE has global time
  - Events have time stamps
- FSM inside DE
  - FSM is a zero delay actor
    - FSM fires whenever DE actor it refines fires
    - Output has same time stamp as input
DE inside FSM

- Analagous to SDF inside FSM
  - DE properties exported to environment of FSM
  - Still have problems with zero-delay loops

Synchronous/Reactive with FSM

- Quick SR system overview
  - Everything happens at ticks
  - Fixed points theorem ensures determinacy
  - Produce phase
    - Function evaluated to determine outputs (find fixed points)
  - Transition phase
    - Outputs changed to prepare for next tick
- Esterel is an example
FSM inside SR

- Simple FSM inside SR
  - Simple => not refined
  - Firing types
    - Type C determines output (if possible)
    - Type D writes the outputs (no calculations)
  - Type C firing occurs until fixed point resolved

FSM inside SR

- Refined FSM inside SR
  - Current state of FSM refined to SR
    - No problem
  - Current state of FSM refined to non-SR
    - Fire these states after SR states (if possible)
    - Invoke transition function (if possible)

- SR inside FSM
  - Pretend FSM is an SR, no problem (just as above)
Verification and Synthesis

- Synthesis from FSM, SDF, SR done before
  - Combining synthesis methods should be possible
- Verification
  - FSM’s and SDF’s separately is well studied
  - Combination unclear, simulation necessary

Example

- Reflex game
  - Heterogenous realization (*charts)
    - Easier to understand
  - Esterel
    - More compact
    - Does not model environment
  - VHDL and C
    - Much longer
    - Messier
- Cell phones
  - Good candidate application for *charts
Conclusion

- *charts
  - Allow designs to use multiple MoC’s
  - Pick MoC most suitable for each part

Happy Halloween!