

Interacting Process Classes

Abhik Roychoudhury
National Univ. of Singapore
Joint work with P.S. Thiagarajan
and Ankit Goel

The problem addressed

- Reactive systems with many similar objects.
 - Telecom -- Phones, Switches.
 - Air-traffic controller -- Incoming aircrafts.
 - Exact # of objects not known at design time.
- Specify and simulate these systems without suffering blow-up.
 - Functional validation of the entire assembly at the design level for a system with large # of (similar) objects.

What kind of objects ?

- Active
 - With a control flow of their own.
 - A process
- Many similar objects in a system
 - A process class
 - Behavior of a process class described via LTS, but the actions are "protocols"
 - A (guarded) Message Sequence Chart is our choice.
 - Description of "protocol" to not central.

Highlights - Simulation

- A symbolic execution semantics for a system with process classes.
 - Leads to space and time efficient simulation of important use cases.
 - State of concrete objects not maintained during execution
 - Name space of objects is not referred
 - Objects grouped into partitions dynamically based on behavior.
 - Partitions are created and merged during simulation.

Highlights - Modeling

- An MOC for reactive systems with many similar processes.
 - A networks of FSMs.
 - One for each "class" of similar processes.
 - Interact on common "communication actions".
 - Communication actions as Sequence Diagrams.
 - The architecture of the system is given by class diagrams.
 - Associations.

Hasn't it been studied ?

- Reasoning about Parameterized Systems
 - Control abstractions for grouping processes, without mentioning their names
- But, associations between processes ??
 - Static
 - Cruiser, Brake control of a car
 - Dynamic
 - Phones engaged in a conversation

Any easy solutions?

- Simulate a system with lesser number of processes per class
 - 10 phones, 20 switches instead of millions
 - How to settle on the cutoff number for exposing all behaviors of the general system
 - Hard for an arbitrary system with complex interactions between processes
 - Results exist for very restricted families of systems – rings etc.

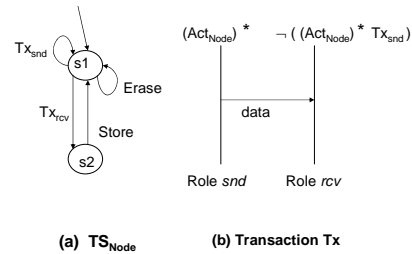
Relevant work

- Live Sequence Charts [Damm/Harel/Marely]
- Symbolic specification of process classes – blow them up during simulation.
- Behavioral Sub-typing
 - [Liskov&Wing, Niestratz,...]
 - Originally studied for passive objects
 - For active obj. use behavioral inclusion from PA
 - Beh. Subclasses, not dynamically changing partitions
- Parameterized Sys. Validation

Organization

- Concrete Execution Semantics
 - Transactions – guarded MSCs
- Symbolic Execution Semantics
 - Dynamically collecting processes of a class into partitions based on their behavior so far
- Checking spurious execution traces
- Examples and Experiments

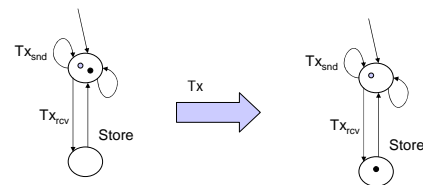
Example of Process Class



Concrete Execution Semantics

- Each action is a transaction
 - Guarded MSC involving several processes.
 - Executed atomically.
- Any execution trace of the system
 - Sequence of MSCs.
 - Synchronous Concatenation
- Guard of a MSC
 - Locally evaluated per-process.

Concrete Execution

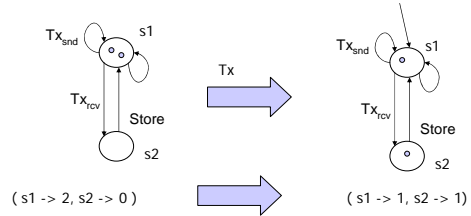


*The number of nodes could be very large, consider
 10^7 phones in a region of a telecom network*

Symbolic Execution Semantics

- States of concrete processes not directly represented
- Partition concrete processes of a process class
 - Current control state
 - History of MSCs executed
 - different histories may lead to diff. futures from the same control state.
- Maintain # of processes in each partition
 - No need to maintain identifiers of behaviorally indistinguishable processes.

Symbolic Simulation



More details due to handling of guards of Tx – not shown.

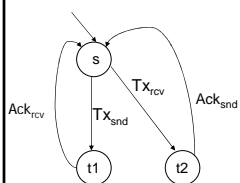
Behavioral Partitions

- Group processes of a class into "Behavioral Partitions"
 - Control state of Process class's FSM.
 - Automata states for each history-based guard of the process class.
- Maintain count of processes in each behavioral partition during simulation.
 - Process names not mentioned anywhere.

The problem now is ...

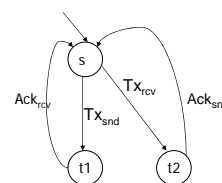
- Put two processes in the same partition
 - When their computation trees are same ...
 - Strong kind of behavioral equivalence ☺
 - Just maintain count of processes in each partition during simulation
 - Counts change as the simulation proceeds ☺
 - How to maintain assoc. between processes
 - System state does **not** refer to process names ☹

Associations



Dynamic relation Wait-for-Ack
 Tx inserts (snd, rcv) to **Wait-for-Ack**
 Ack checks (rcv, snd) ∈ **Wait-for-Ack**
 Ack deletes (rcv, snd) from **Wait-for-Ack**

Simulation



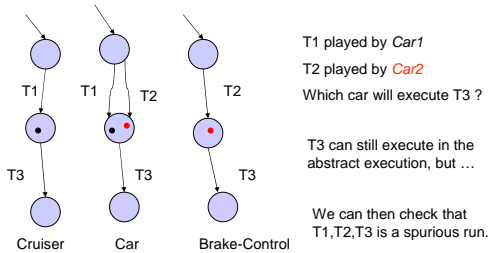
Initially, s -> 100, t1 -> 0, t2 -> 0

Execute Tx

s -> 98, t1 -> 1, t2 -> 1
 In addition, maintain
 Wait-for-ack = { (t1, t2) }

Maintain associations between behavioral partitions

Abstract exec. allows spurious traces



Checking simulation runs

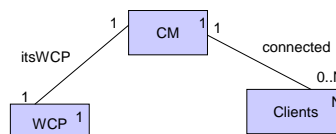
- Decidable in our control abstraction setting.
 - Process classes may contain unbounded objects.
 - For predicate abstraction of data vars., accumulate constraints from trace – constraints can refer to any operation appearing in the program.
- To check run $\sigma = T_1, \dots, T_k$
 - ensure that σ is a concrete run in a sys. where a process class p has at least $X(p, \sigma)$ objects.
 - $X(p, \sigma) =$
 - total # of times p occurs in transactions T_i of σ
- More efficient procedure via eq. constraint solving exists.

Modeled Examples

NASA CTAS

- Automation tools for managing large volume arrival air traffic in large airports.
- Final Approach Spacing Tool
 - Determine speed and trajectory of incoming aircrafts on their final approach.
 - Master controller updates weather info. to "clients"
 - controllers using inputs to compute aircraft trajectories.
 - Modeled and simulated the Weather update subsystem from Requirements Document.

Modeled Examples



- WCP -- Weather Control Panel
 (contains weather info.)
- CM -- Communications Manager
 (transfers info from WCP to clients)
- Clients -- Weather aware, seek connection with CM
- No need to maintain clients' names during simulation.

Experience

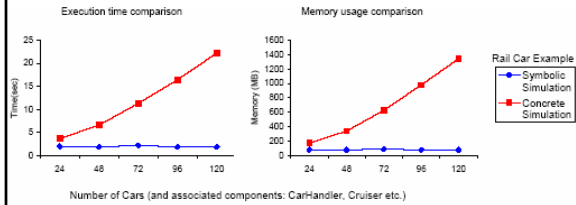
- Cuts simulation time/memory for diff. controllers
 - CTAS weather update controller
 - Rail Shuttle system from Paderborn
 - Examples for State + Seq. Diagram based modeling
 - Rail car (from Live Sequence Charts – modeled in our MOC)
 - Many cars operating in two parallel cyclic paths
 - Complex System, many process classes + assoc.
 - cars, cruisers, proximity sensors, Terminal, Carhandler ...
 - Telephone switch network (from SPIN's benchmark suite)
 - Call-waiting, 3-way calling and other features in tel. network
- Simulator found realizable bugs in the examples
 - Deadlock scenarios in CTAS weather controller

Simulation Results

	Time (C) secs	Time (S) secs	Mem (C) MB	Mem (S) MB
Rail-car (24 cars)	2.1	3.9	83	173
Rail-car (48 cars)	2.2	7.0	84	153
Shuttle (30)	0.44	0.7	18	33
Shuttle (60)	0.44	1.2	18	69
CTAS (10)	1.5	2	63	87
CTAS (20)	1.5	4.1	64	189

Simulation stopped after 1000 transactions

Symbolic vs concrete exec



Wrapping up

- Combining intra-component and inter-component style to produce an executable spec.
- Avoiding blow-up in specification and execution of such specs. due to **many** similar processes.
 - Symbolic execution semantics
 - Can check whether a exec run corresponds to a concrete one.
- Many other challenging issues
 - Hierarchy of process classes?
 - We only consider a collection of process classes
 - Abstraction refinement based Model Checker?