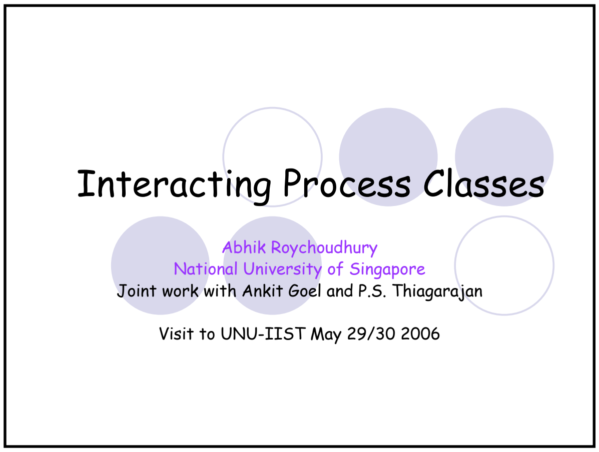
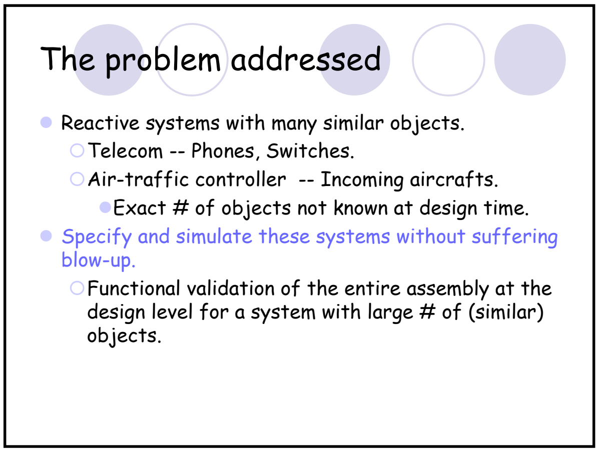


Interacting Process Classes

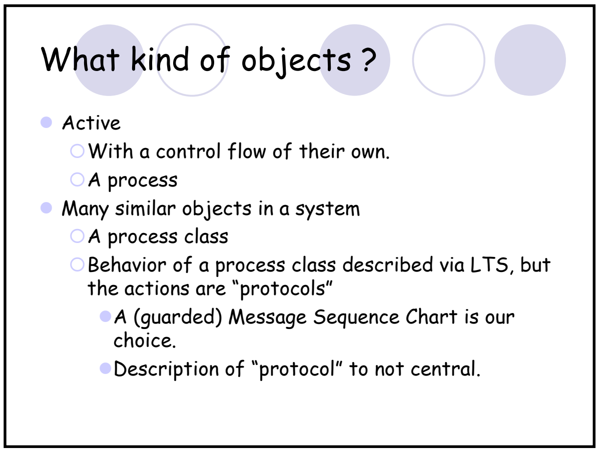
Abhik Roychoudhury
National University of Singapore
Joint work with Ankit Goel and P.S. Thiagarajan

Visit to UNU-IIST May 29/30 2006



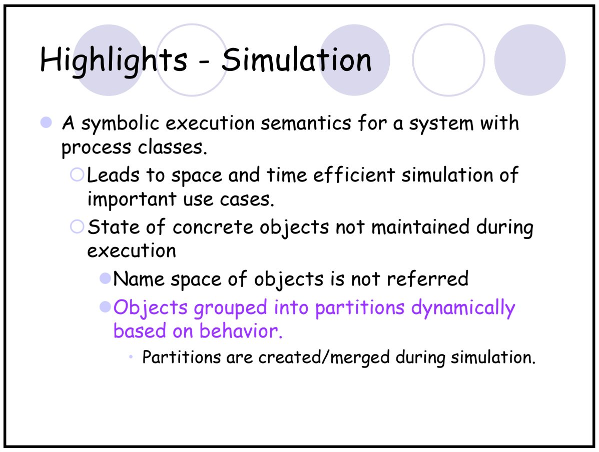
The problem addressed

- 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.



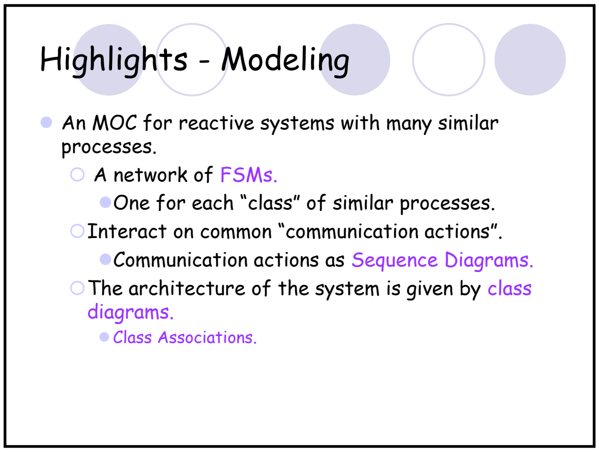
What kind of objects ?

- 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.



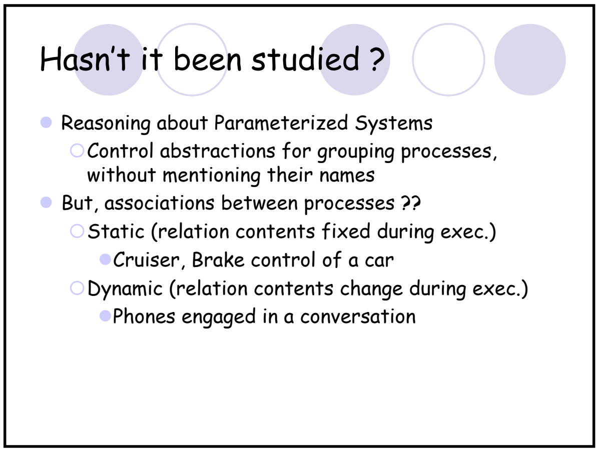
Highlights - Simulation

- 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/merged during simulation.



Highlights - Modeling

- An MOC for reactive systems with many similar processes.
  - A network 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.
    - Class Associations.



Hasn't it been studied ?

- Reasoning about Parameterized Systems
  - Control abstractions for grouping processes, without mentioning their names
- But, associations between processes ??
  - Static (relation contents fixed during exec.)
    - Cruiser, Brake control of a car
  - Dynamic (relation contents change during exec.)
    - Phones engaged in a conversation

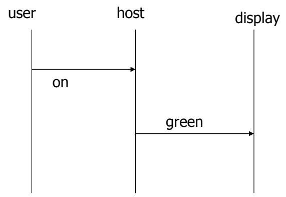
Any easy solutions?

- 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

- Executable MSC based modeling languages
  - Live Sequence Chart [Damm/Harel/Marelly] is such an effort
  - Only 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 System Validation: already discussed

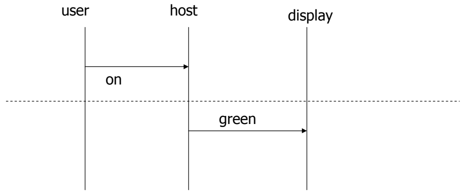
MSCs --- Possible Behavior



The user may switch on the host following by the host turning the display to green.

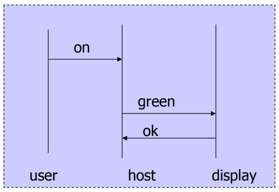
Illegal Behavior

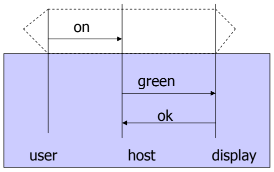
How to specify such requirements?



Whenever the user switches on the host , the host must turn the display to green.

Live Sequence Charts





Existential Chart

Universal Chart

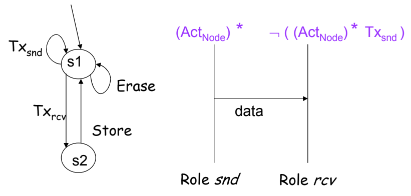
View from MSC angle

- Message Sequence Charts
  - Scenarios in Sys. Execution --- Weak form of Requirement
  - Says what is possible, not what is not possible.
- Live Sequence Charts
  - Executable Requirements based on MSCs
  - Centralized execution semantics which monitors all charts in the specification.
  - No support for Process Classes
- Interacting Process Classes
  - Executable, with per-process semantics.
  - Symbolic execution semantics to support classes.

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 a Process Class



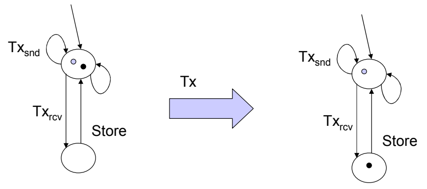
TS_{Node}

Transaction Tx

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



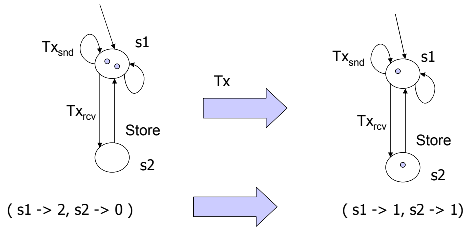
The number of nodes could be very large, consider

10^7 phones in a region of a telecom network

Symbolic Execution Semantics

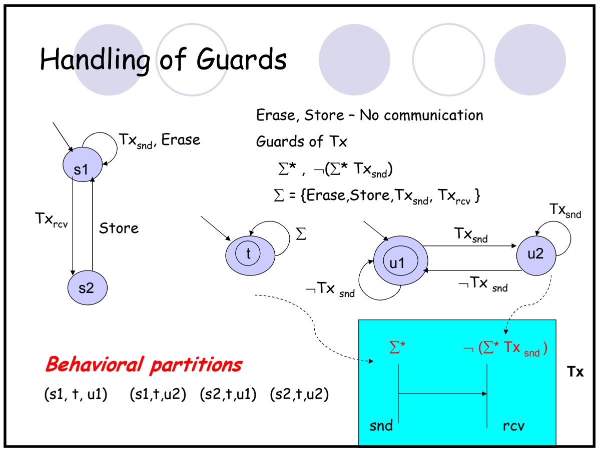
- States of concrete processes not directly represented
- Partition concrete processes of a process class
  - Current control state
  - History of MSCs executed
    - different histories may lead to diff. futures from the same control state.
    - Regular expression over MSC alphabet.
- Maintain # of 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.

Handling of Guards



Erase, Store – No communication

Guards of Tx

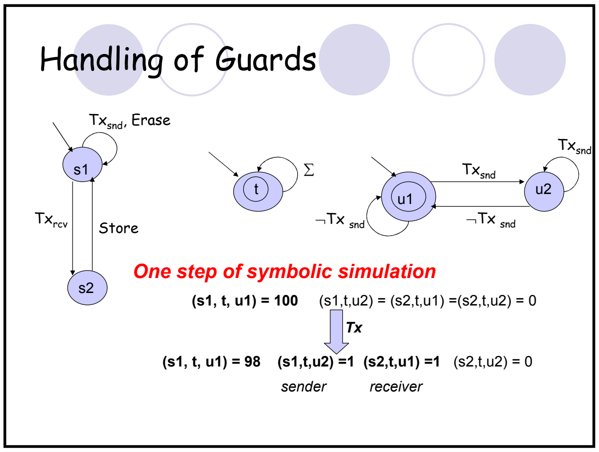
Σ^* , $\neg(\Sigma^*\ Tx_{snd})$

$\Sigma = \{Erase, Store, Tx_{snd}, Tx_{rcv}\}$

Behavioral partitions

(s1, t, u1) (s1,t,u2) (s2,t,u1) (s2,t,u2)

Handling of Guards



One step of symbolic simulation

(s1, t, u1) = 100 (s1,t,u2) = (s2,t,u1) =(s2,t,u2) = 0

(s1, t, u1) = 98 **(s1,t,u2) =1** **(s2,t,u1) =1** (s2,t,u2) = 0

sender *receiver*

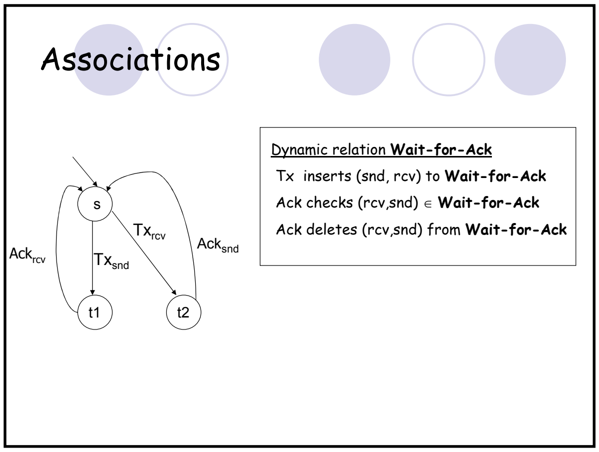
Behavioral Partitions

- 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



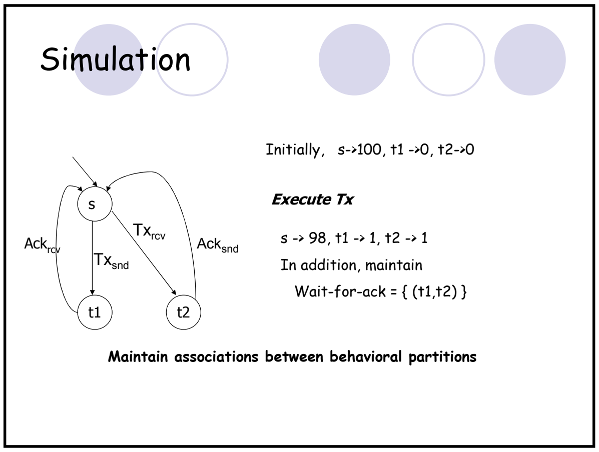
Dynamic relation **Wait-for-Ack**

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

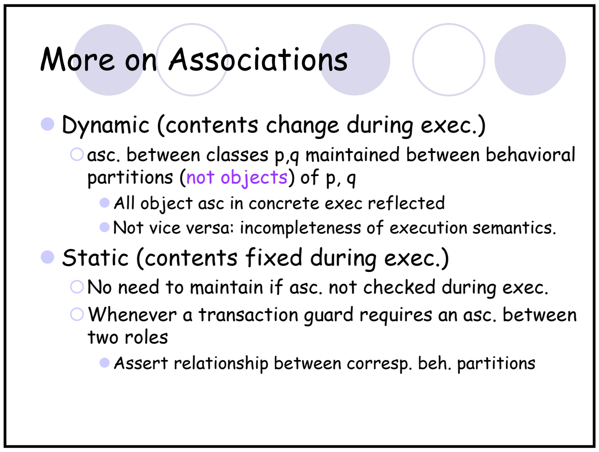
Execute Tx

s -> 98, t1 -> 1, t2 -> 1

In addition, maintain

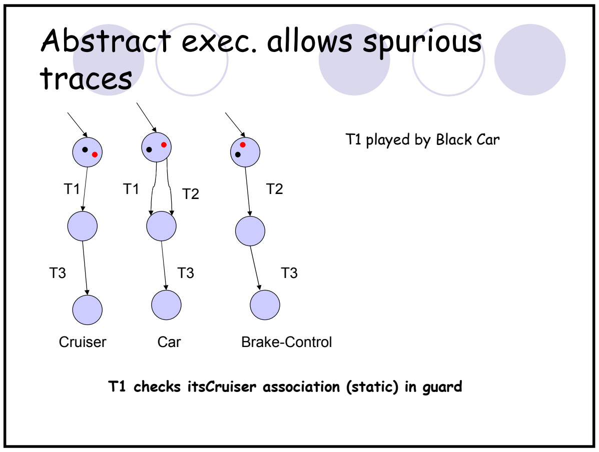
Wait-for-ack = { (t1,t2) }

Maintain associations between behavioral partitions



More on Associations

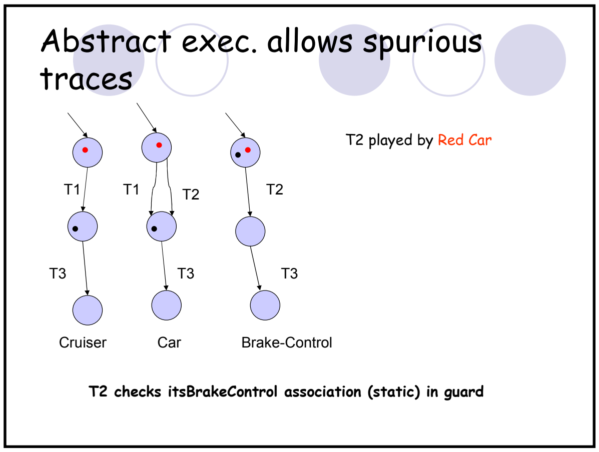
- Dynamic (contents change during exec.)
  - asc. between classes p,q maintained between behavioral partitions (not objects) of p, q
    - All object asc in concrete exec reflected
    - Not vice versa: incompleteness of execution semantics.
- Static (contents fixed during exec.)
  - No need to maintain if asc. not checked during exec.
  - Whenever a transaction guard requires an asc. between two roles
    - Assert relationship between corresp. beh. partitions



Abstract exec. allows spurious traces

T1 played by Black Car

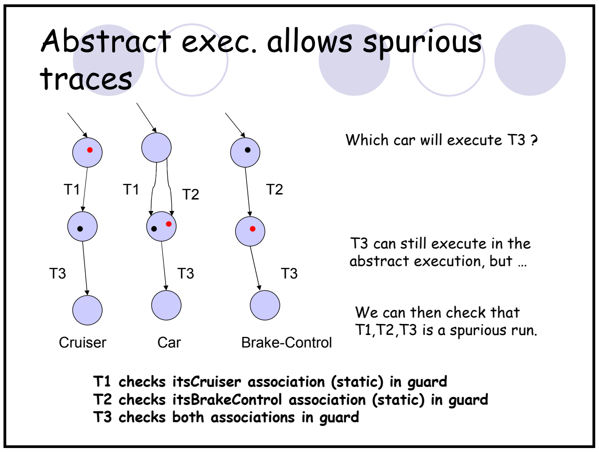
T1 checks itsCruiser association (static) in guard



Abstract exec. allows spurious traces

T2 played by Red Car

T2 checks itsBrakeControl association (static) in guard



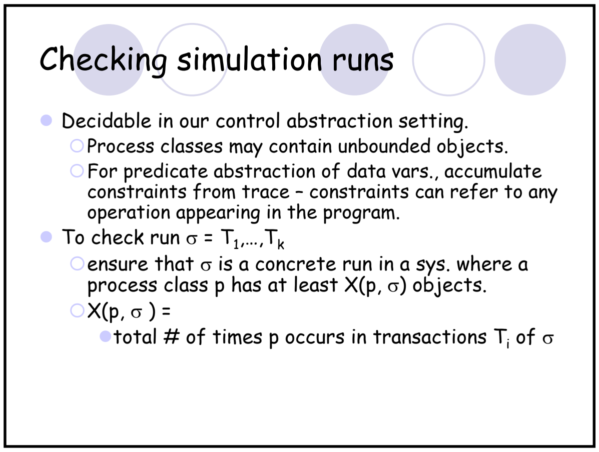
Abstract exec. allows spurious traces

Which car will execute T3 ?

T3 can still execute in the abstract execution, but …

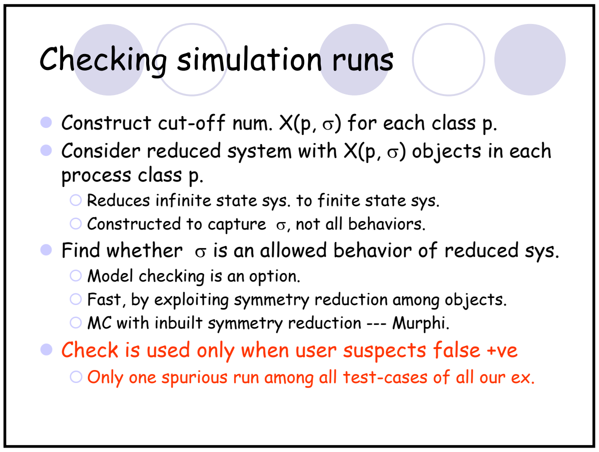
We can then check that T1,T2,T3 is a spurious run.

T1 checks itsCruiser association (static) in guard
T2 checks itsBrakeControl association (static) in guard
T3 checks both associations in guard



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 σ = $T_1,\ldots,T_k$
  - ensure that σ is a concrete run in a sys. where a process class p has at least X(p, σ) objects.
  - X(p, σ) =
    - total # of times p occurs in transactions T_i of σ



Checking simulation runs

- Construct cut-off num. X(p, σ) for each class p.
- Consider reduced system with X(p, σ) objects in each process class p.
  - Reduces infinite state sys. to finite state sys.
  - Constructed to capture σ, not all behaviors.
- Find whether σ is an allowed behavior of reduced sys.
  - Model checking is an option.
  - Fast, by exploiting symmetry reduction among objects.
  - MC with inbuilt symmetry reduction --- Murphi.
- Check is used only when user suspects false +ve
  - Only one spurious run among all test-cases of all our ex.

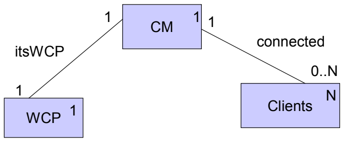
What do we have ?

But, the proof of the pudding is…

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.**

Weather update Subsystem



WCP -- Weather Control Panel

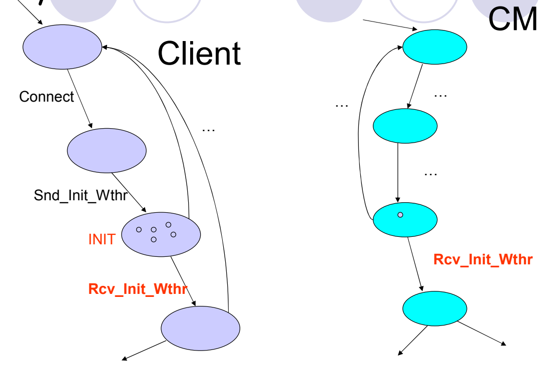
(contains weather info.)

CM -- Communications Manager

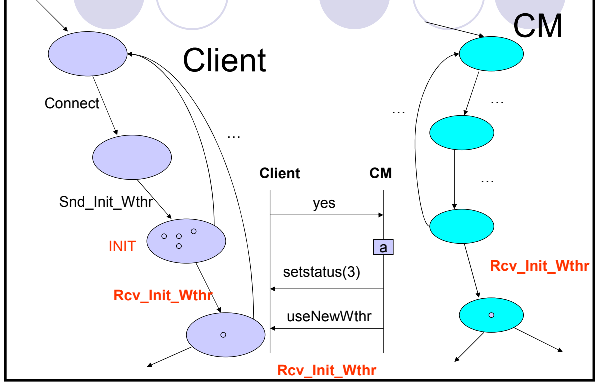
(transfers info from WCP to clients)

Clients – Weather aware, seek connection with CM

Symbolic Simulation

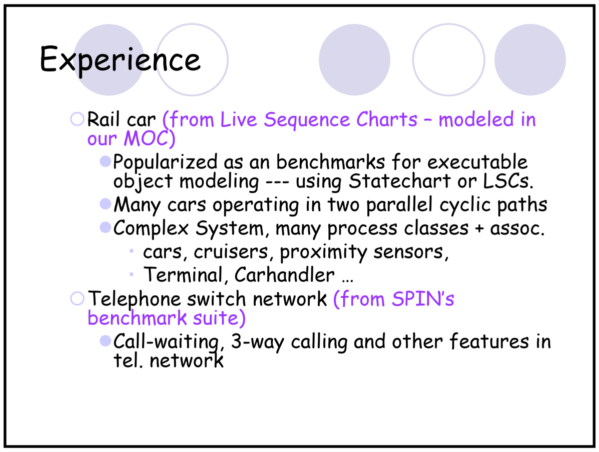


Symbolic Simulation



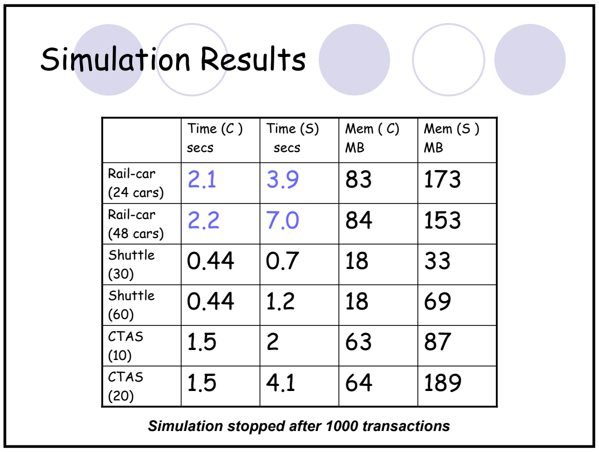
Experience

- Cuts simulation time/memory for diff. controllers
  - CTAS weather update controller
    - Simulator found realizable bugs in the examples
    - Deadlock scenarios in CTAS weather controller
  - Rail Shuttle system from Paderborn
    - **Examples for State + Seq. Diagram based modeling**
    - **Controller for a rail shuttle system where shuttles bid for orders to transport passengers in an interconnection network.**



Experience

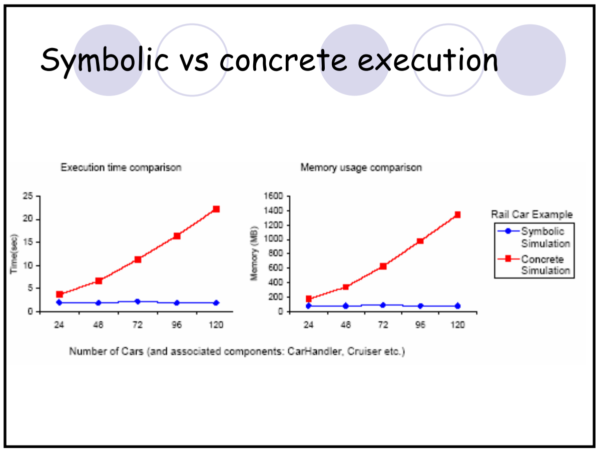
- Rail car (from Live Sequence Charts – modeled in our MOC)
  - Popularized as an benchmarks for executable object modeling --- using Statechart or LSCs.
  - Many cars operating in two parallel cyclic paths
  - Complex System, many process classes + assoc.
    - cars, cruisers, proximity sensors,
    - Terminal, Carhandler …
- Telephone switch network (from SPIN's benchmark suite)
  - Call-waiting, 3-way calling and other features in tel. network



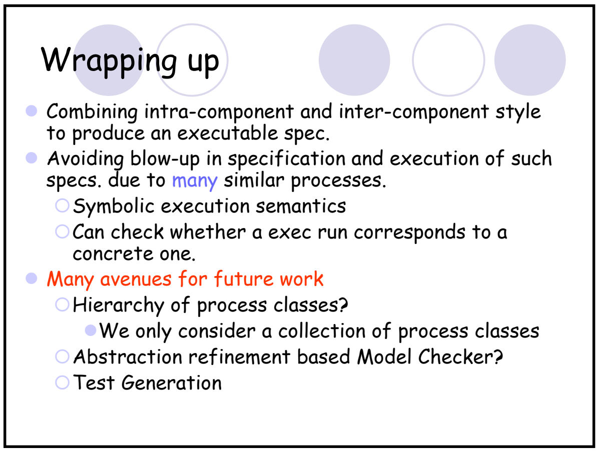
Simulation Results

	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

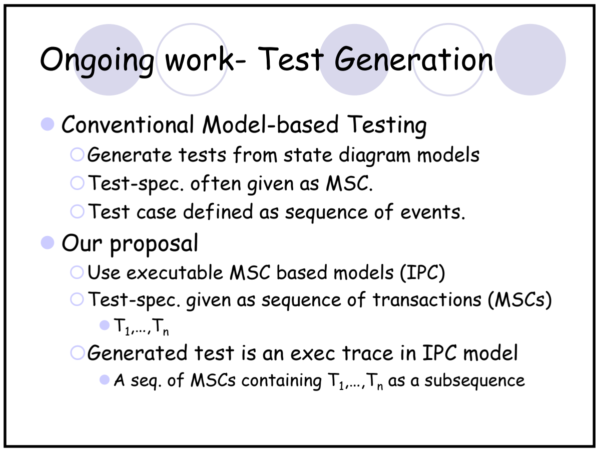


Symbolic vs concrete execution



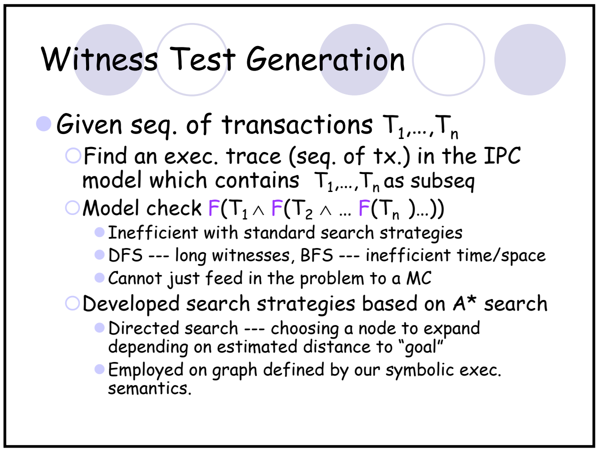
Wrapping up

- 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 avenues for future work
  - Hierarchy of process classes?
    - We only consider a collection of process classes
  - Abstraction refinement based Model Checker?
  - Test Generation



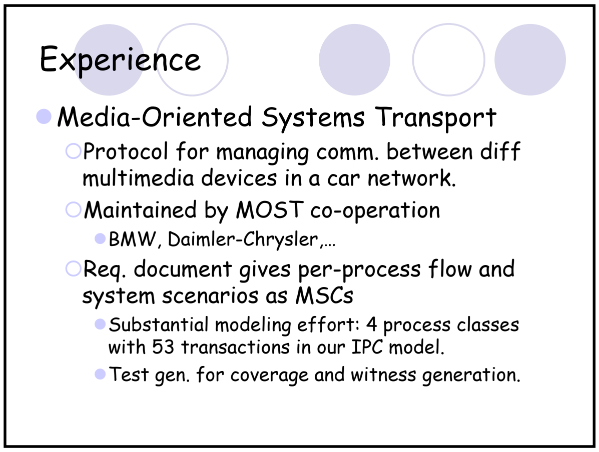
Ongoing work- Test Generation

- Conventional Model-based Testing
  - Generate tests from state diagram models
  - Test-spec. often given as MSC.
  - Test case defined as sequence of events.
- Our proposal
  - Use executable MSC based models (IPC)
  - Test-spec. given as sequence of transactions (MSCs)
    - $T_1,\ldots,T_n$
  - Generated test is an exec trace in IPC model
    - A seq. of MSCs containing $T_1,\ldots,T_n$ as a subsequence



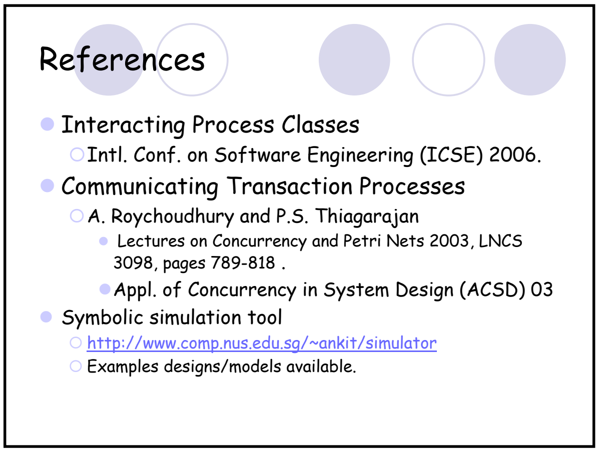
Witness Test Generation

- Given seq. of transactions $T_1,\ldots,T_n$
  - Find an exec. trace (seq. of tx.) in the IPC model which contains $T_1,\ldots,T_n$ as subseq
  - Model check $F(T_1 \wedge F(T_2 \wedge \ldots F(T_n)\ldots))$
    - Inefficient with standard search strategies
    - DFS --- long witnesses, BFS --- inefficient time/space
    - Cannot just feed in the problem to a MC
  - Developed search strategies based on A* search
    - Directed search --- choosing a node to expand depending on estimated distance to "goal"
    - Employed on graph defined by our symbolic exec. semantics.



Experience

- Media-Oriented Systems Transport
  - Protocol for managing comm. between diff multimedia devices in a car network.
  - Maintained by MOST co-operation
    - BMW, Daimler-Chrysler,…
  - Req. document gives per-process flow and system scenarios as MSCs
    - Substantial modeling effort: 4 process classes with 53 transactions in our IPC model.
    - Test gen. for coverage and witness generation.



References

- Interacting Process Classes
  - Intl. Conf. on Software Engineering (ICSE) 2006.
- Communicating Transaction Processes
  - A. Roychoudhury and P.S. Thiagarajan
    - Lectures on Concurrency and Petri Nets 2003, LNCS 3098, pages 789-818 .
    - Appl. of Concurrency in System Design (ACSD) 03
- Symbolic simulation tool
  - http://www.comp.nus.edu.sg/~ankit/simulator
  - Examples designs/models available.