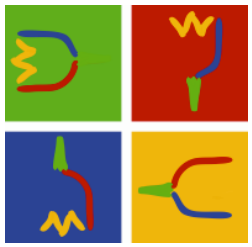
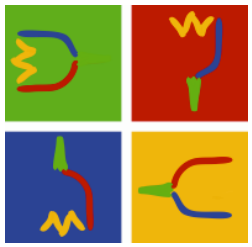


A Process Modeling Framework for Formal Validation of Panama Canal System Operations

Mark Austin and John Johnson,

Department of Civil Engineering and ISR,

University of Maryland, College Park, MD 20742.

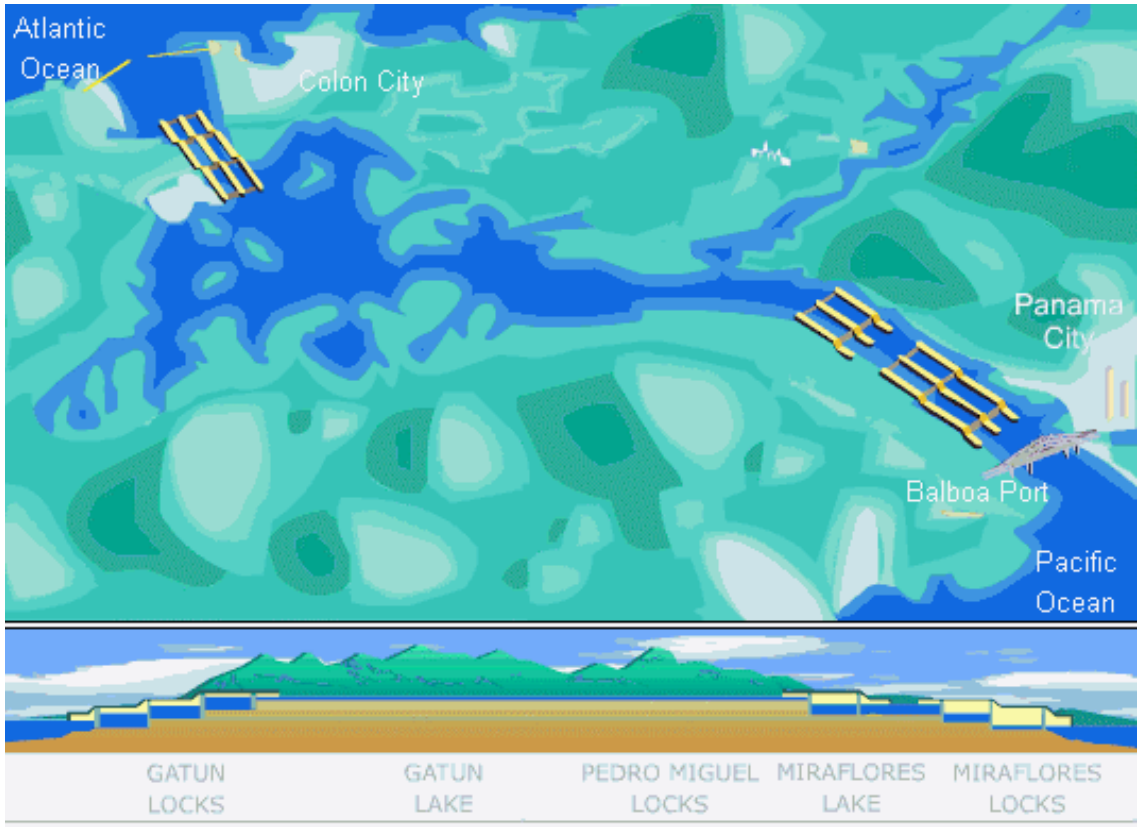


Outline

- Motivation
- Early Validation of Systems
- Automated Composition of Process Models
- LTSA
- Lockset Process Model
- Model Checking
- Conclusions/Future Work

Motivation

The Panama Canal has been operational since 1914.



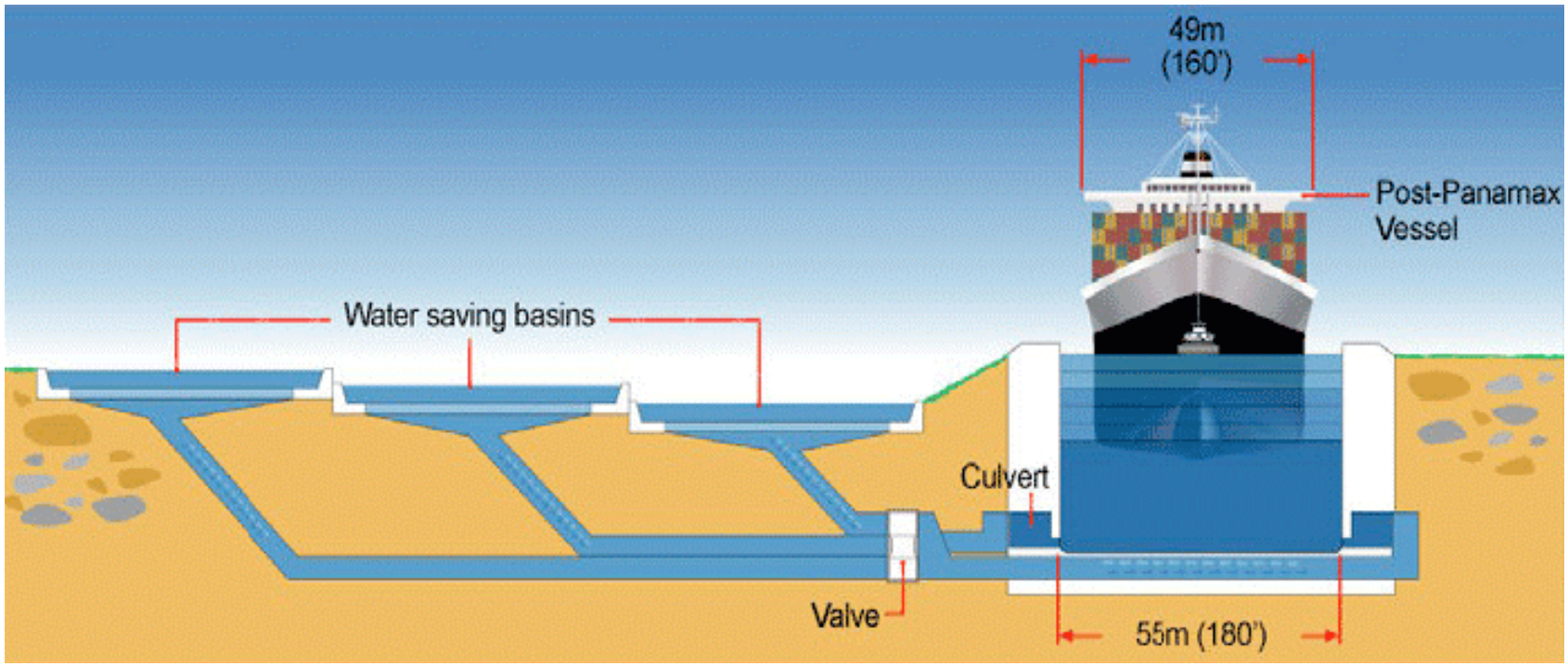
Motivation

Left: Congestion in the Miraflores lock and lake.

Right: Limitations of present-day canal capacity…

Motivation

The Panama Canal is currently undergoing a US $5.25 billion renovation.

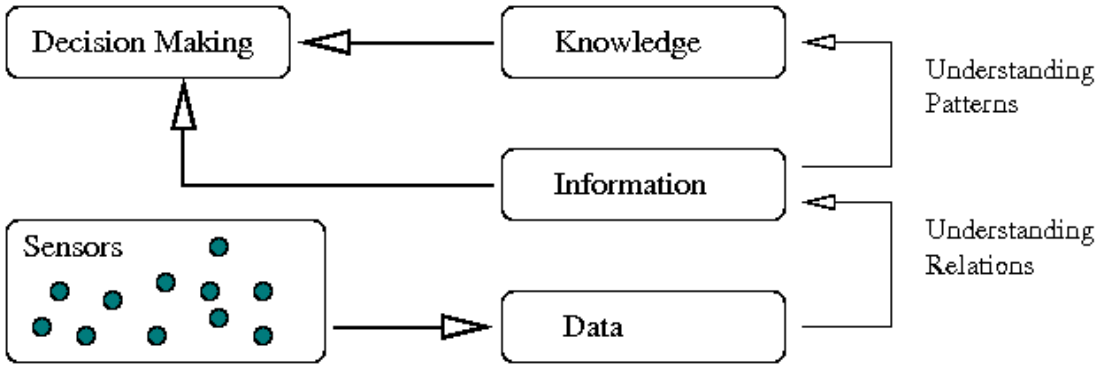


Cross section of water saving basins and a laterally filled lock chamber

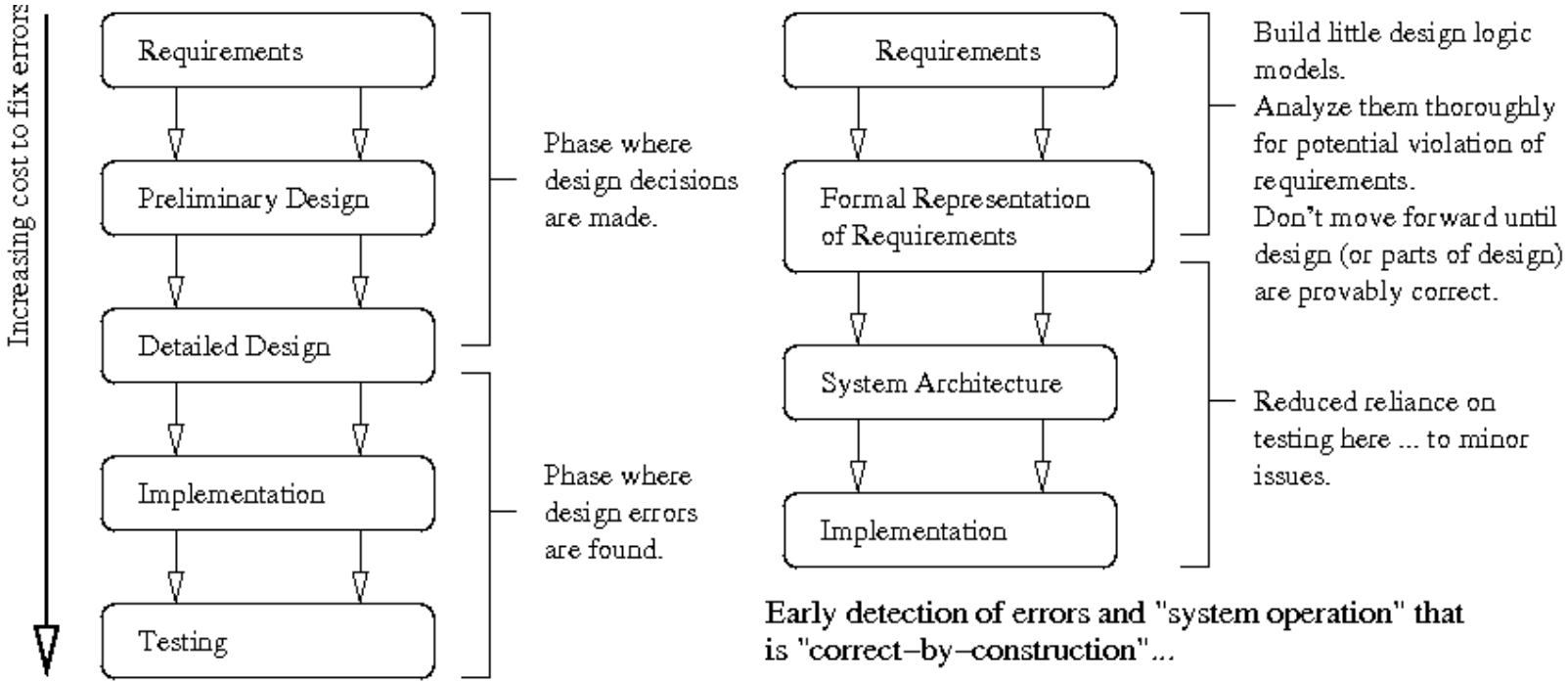
SE Education at UMCP

Industrial-Age Systems	Information-Age Systems
Small, simple, Linear	Large, complex, nonlinear
Systems of Components	System of systems
Dominated by hardware	Combinations of hardware, software and communications

Many present-day systems are limited in their situation awareness and ability to look ahead and predict events.



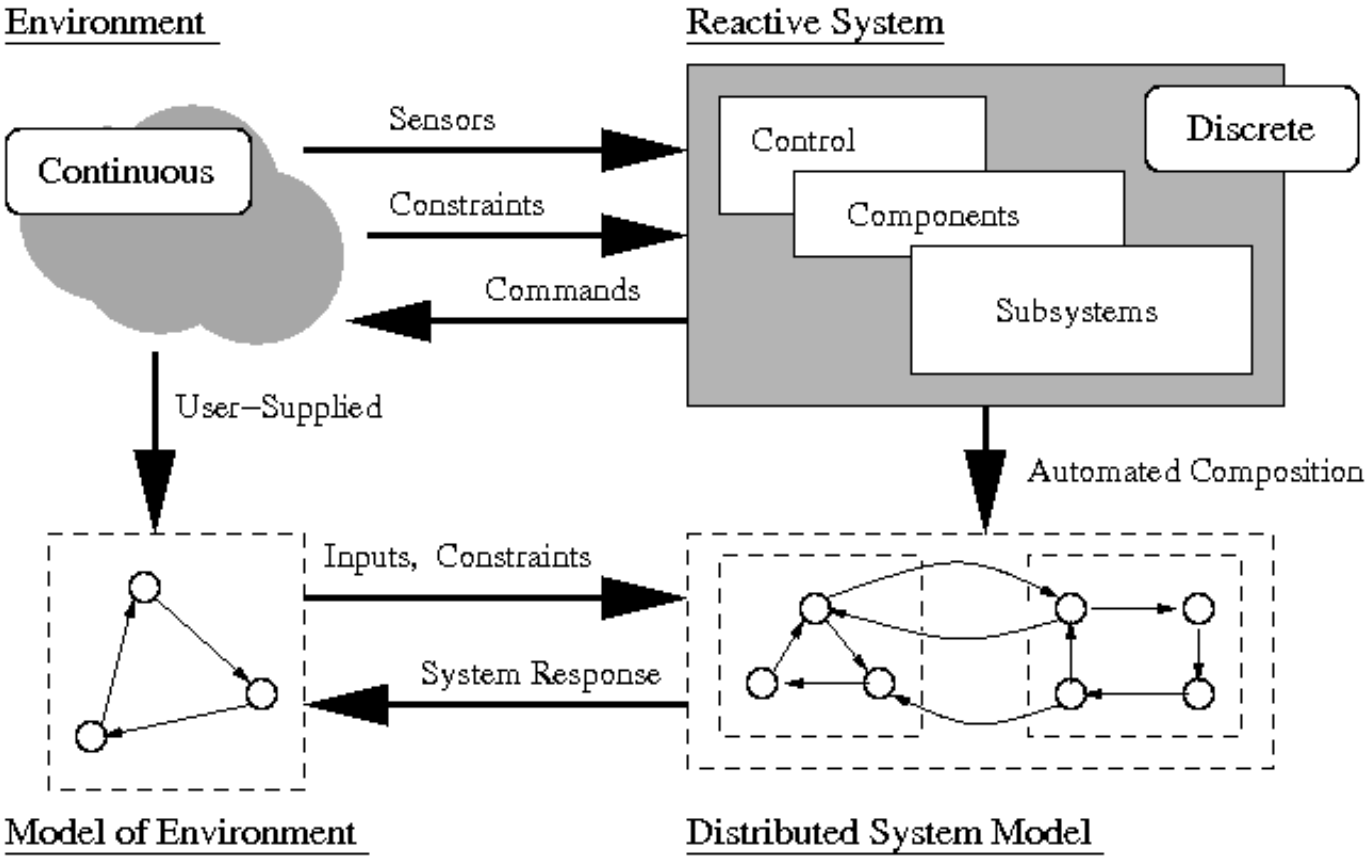
Early Validation of Systems

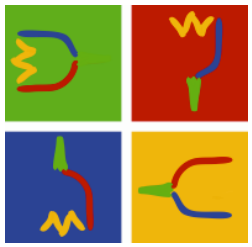


Traditional Approach to Airportal Design and Test.

Early detection of errors and "system operation" that is "correct–by–construction"...

Composition of Models





What is LTSA?

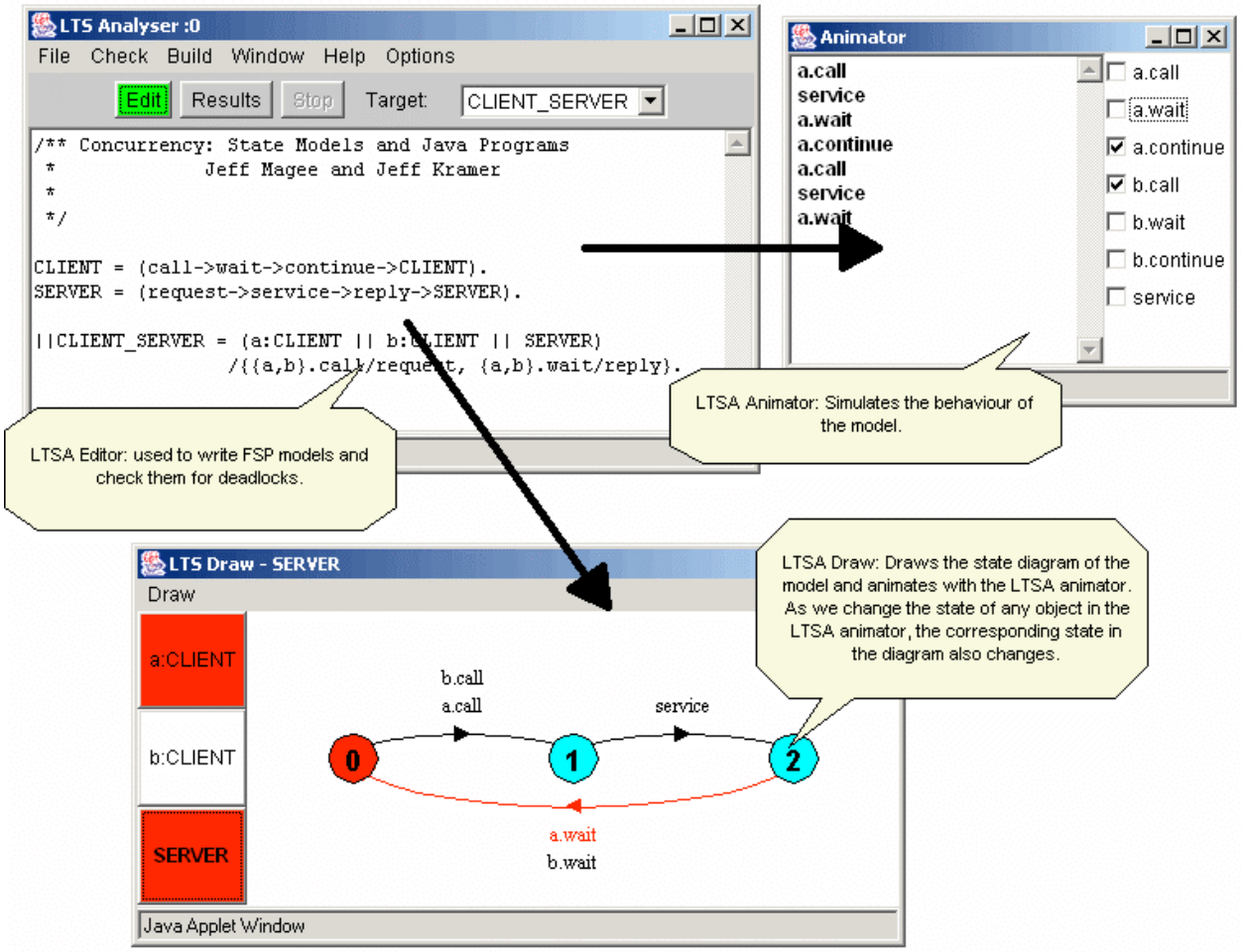
The labeled transition system analyzer (LTSA) is…

> **verification tool for concurrent systems. It mechanically checks that the specification of a concurrent system satisfies the properties required of its behavior.**

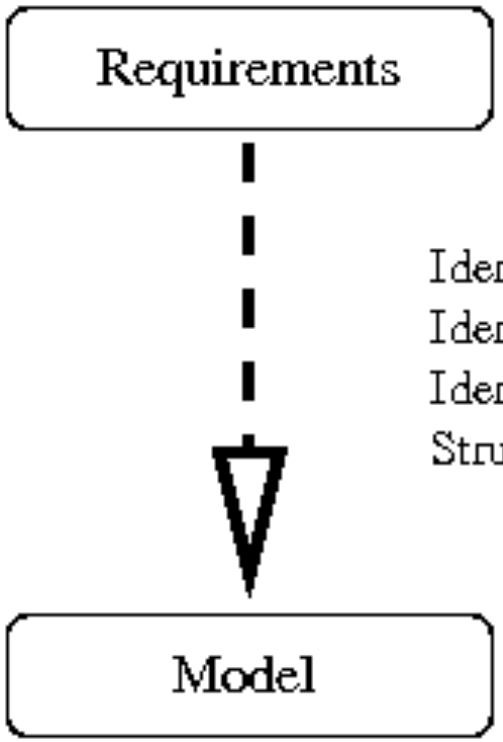
It is particularly suitable for high-level modeling and verification of systems dominated by processes that have concurrent behaviors, including interaction with other processes.

LTSA supports specification animation to facilitate interactive exploration of system behavior.

Working with LTSA



From requirements to architectures….



Goals of the system...
Scenarios / Use case models....
Properties of interest ....

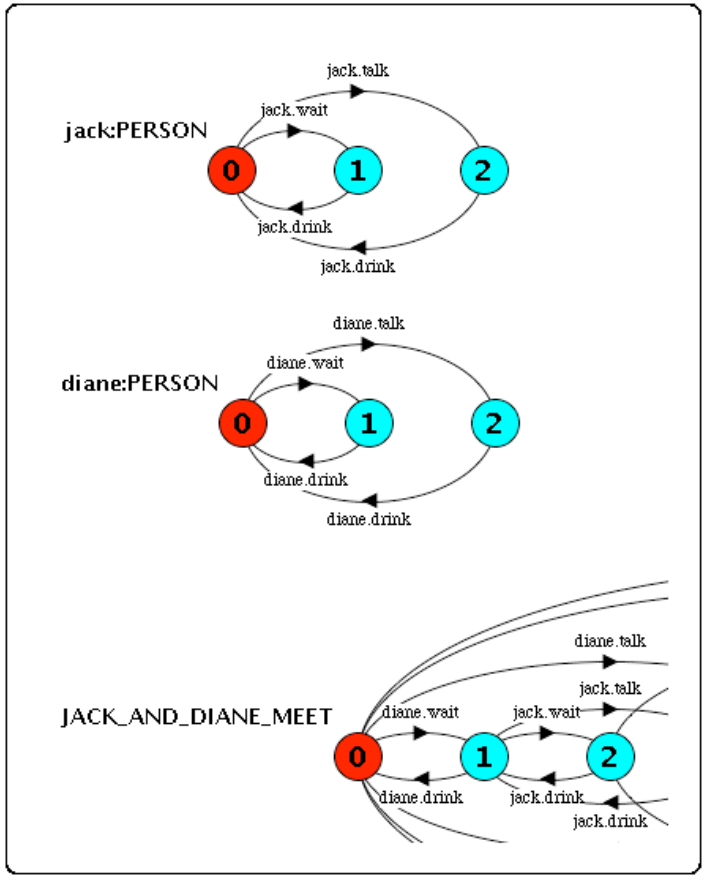
Identify main events, actions, and interactions ....
Identify and define main processes ...
Identify and define properties of interest ...
Structure processes into an architecture

Check traces of interest ...
Check properties of interest ....

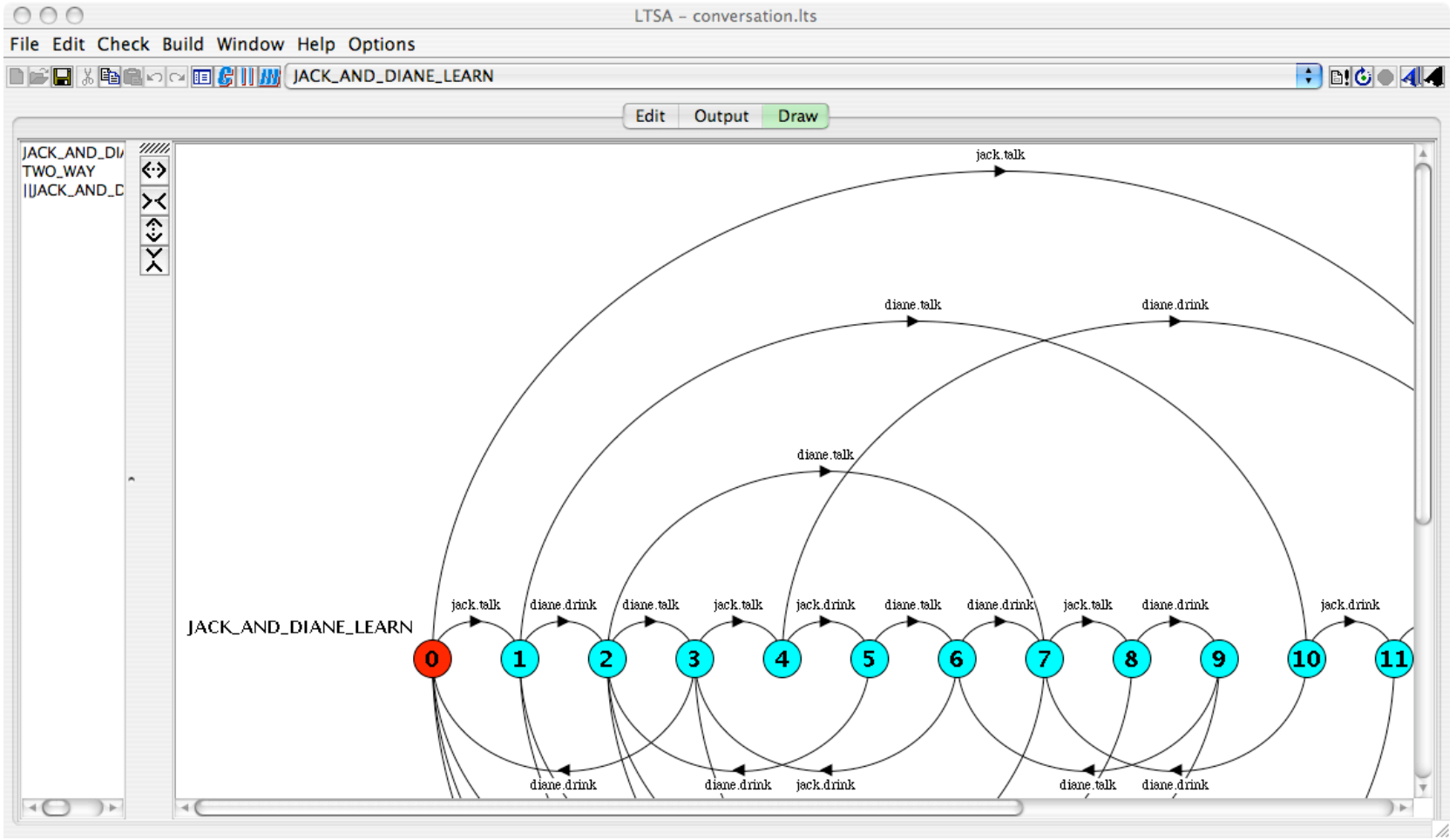
A Simple Example

Two friends talk over coffee …..

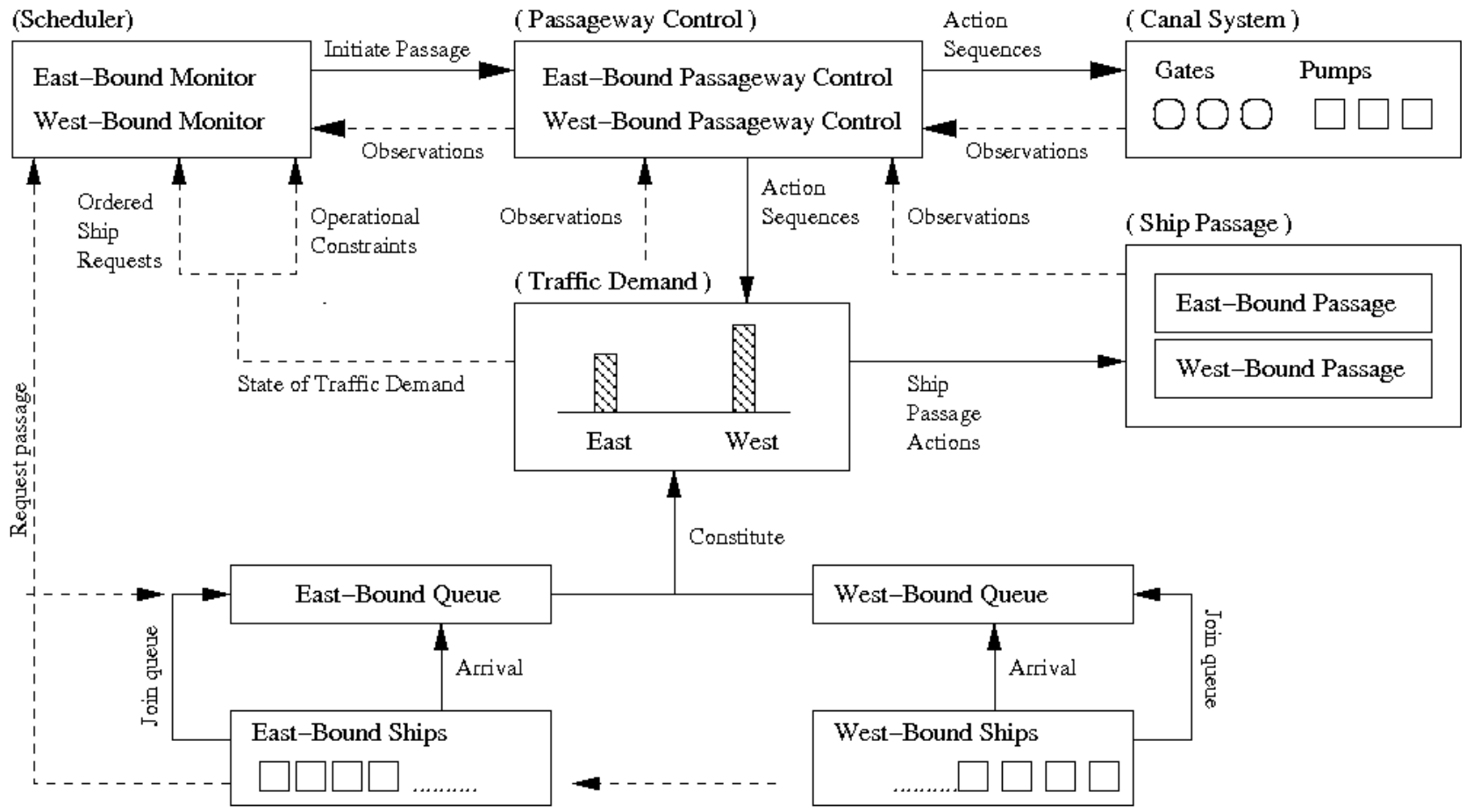
```
Terminal — tcsh — 70x27
/Users/austin/ltsa3.0/Austin 277>> more conversation.lts
// ======================================================================
// Jack and Diane have conversation over coffee ....
// ======================================================================

// Create a person who: (1) talks and drinks coffee, or
//                      (2) just waits and then drinks coffee ....

   PERSON = (   talk -> drink -> PERSON
              | wait -> drink -> PERSON ).

// Jack and Diane meet ....

 ||JACK_AND_DIANE_MEET = ( jack:PERSON || diane:PERSON ).

// To learn, conversation needs to be two way ....

   TWO_WAY = ( jack.talk -> diane.talk -> TWO_WAY ).

// Conversation should be polite ....

 ||JACK_AND_DIANE_LEARN = ( JACK_AND_DIANE_MEET || TWO_WAY ) / {
                     jack.talk/diane.wait, diane.talk/jack.wait }.

// ======================================================================
// End!
/Users/austin/ltsa3.0/Austin 278>>
```



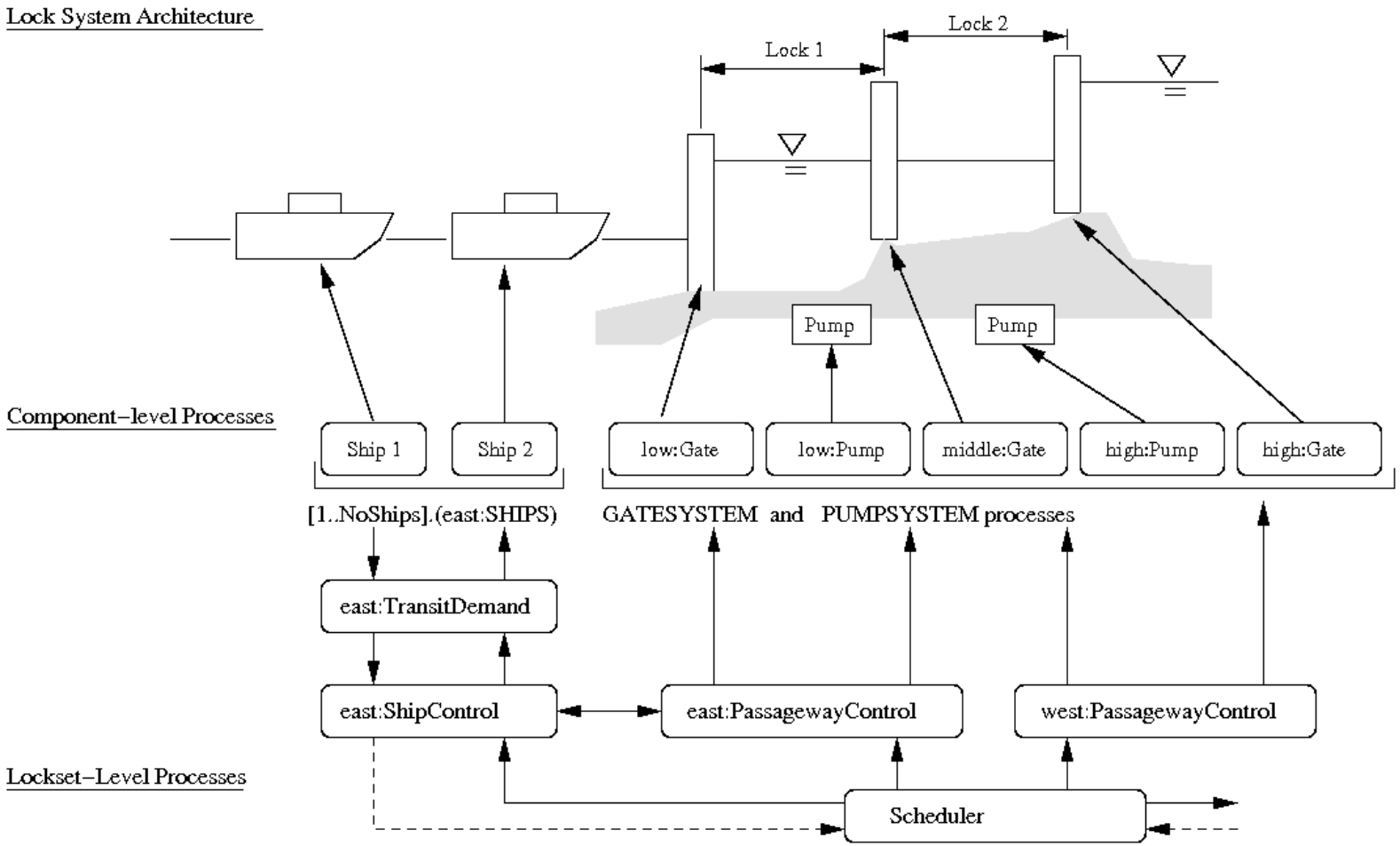
Jack and Diane Talk!!



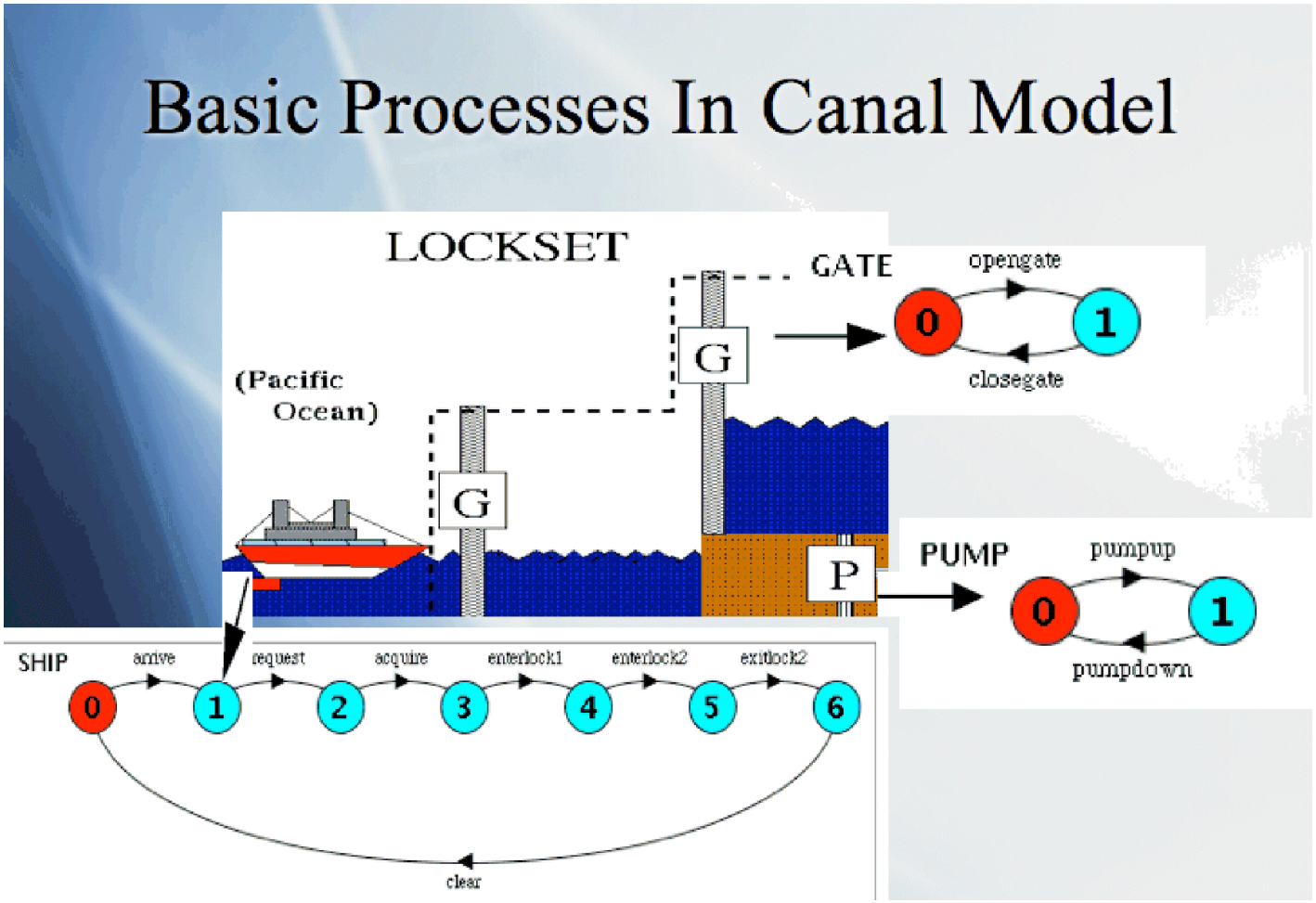
Framework for Model Development



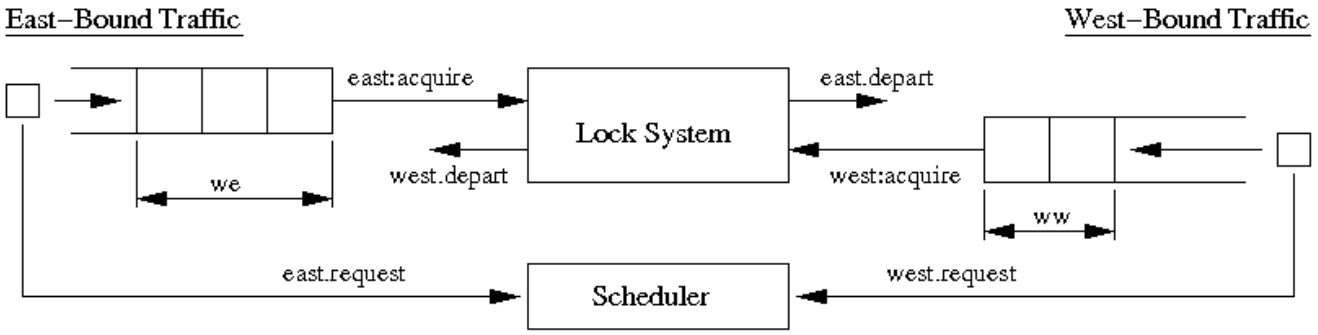
Lockset Process Model

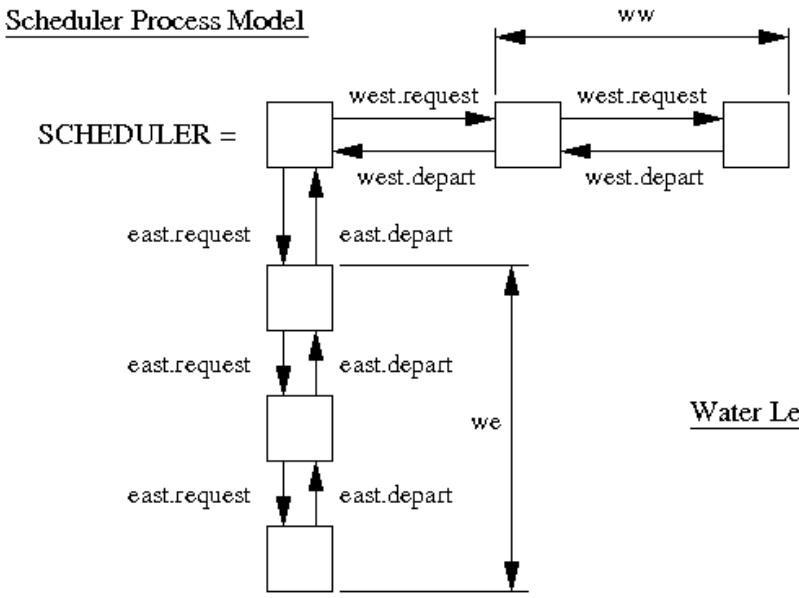


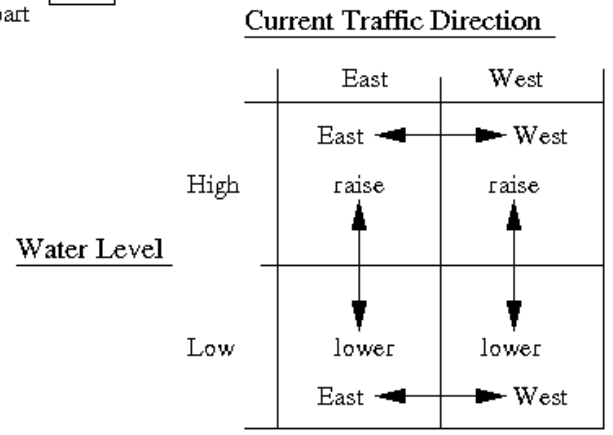
Component Processes

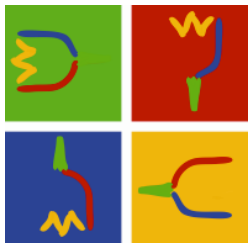


Scheduler Design







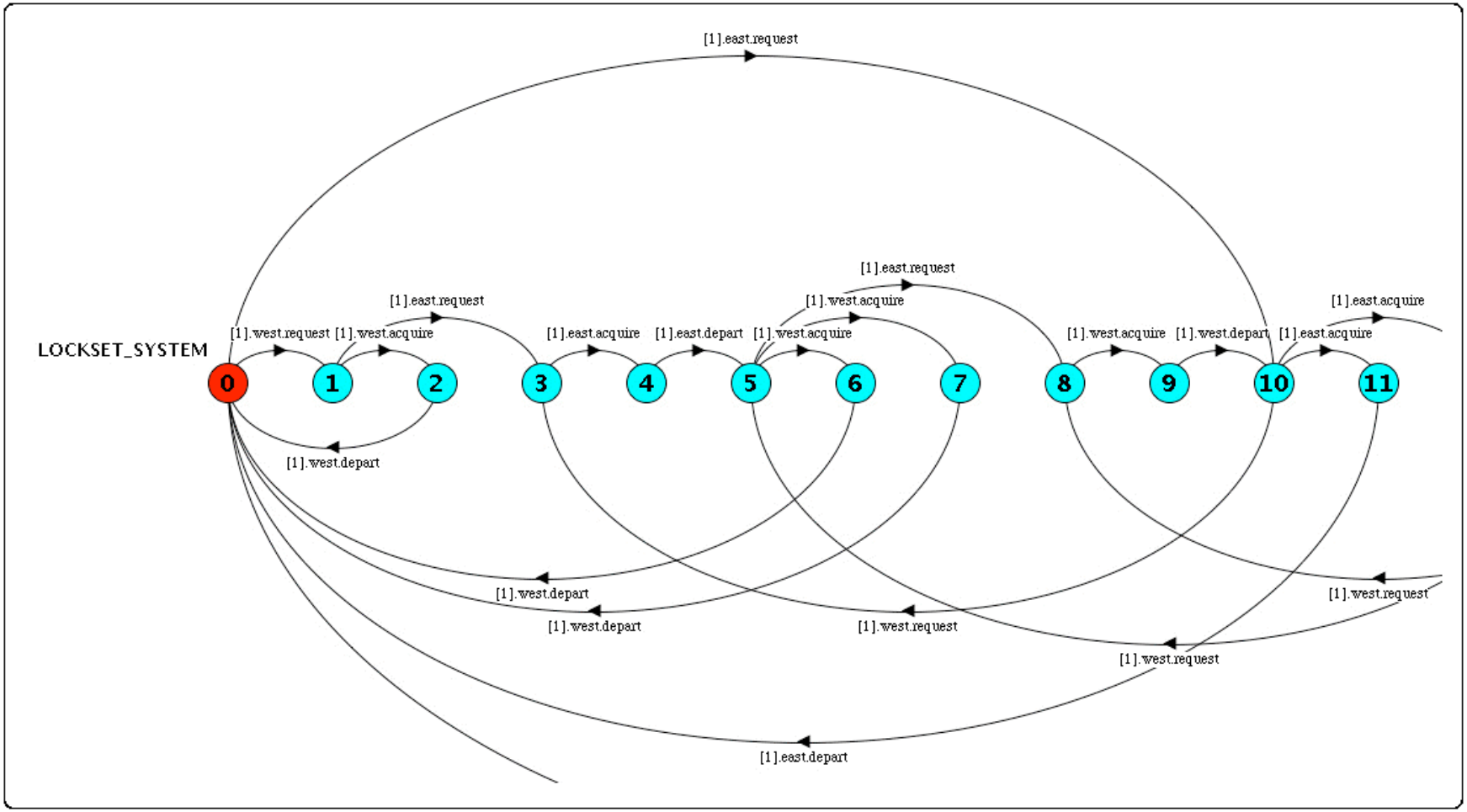


Scheduler Design

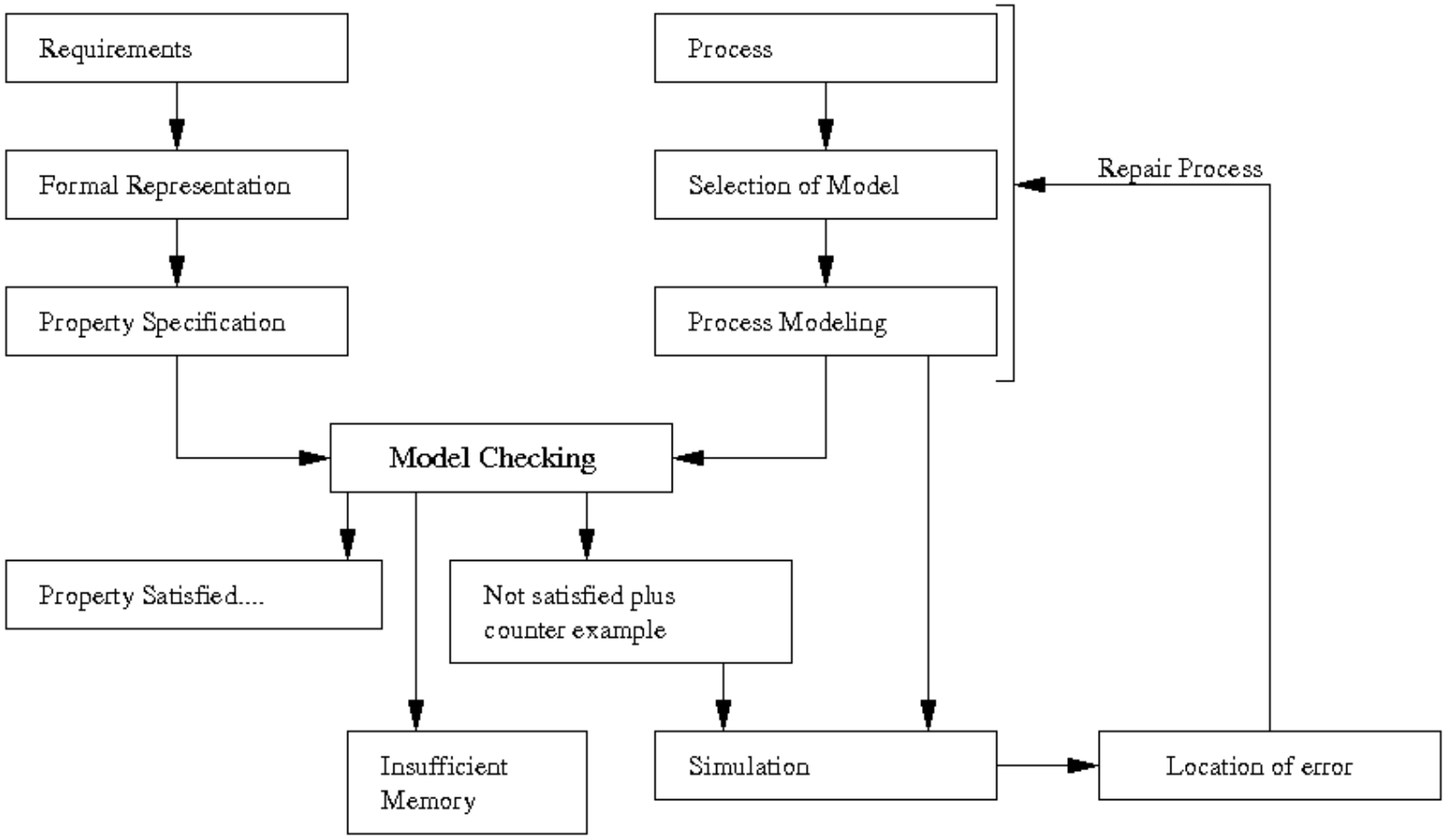
Terminal — vim — 93x22

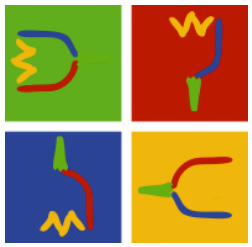
```
SCHEDULER = SCHEDULER[0][0][East][Low],
SCHEDULER[we:0..NoShips][ww:0..NoShips][td:TrafficDirection][wl:WaterLevel] = (

  // Register requests to transit lock in east- and west-bound directions.

      when ( we <= NoShips ) [S].east.request -> SCHEDULER[we+1][  ww][td][wl]
    | when ( ww <= NoShips ) [S].west.request -> SCHEDULER[  we][ww+1][td][wl]

  // East-bound assignments to ascend the lock system.

    | when ( we >= 1 && td == East && wl == Low )  [i:S].east.acquire ->
             ascend   -> [i].east.depart -> SCHEDULER[we-1][  ww][West][wl]
    | when ( ww == 0 && we >= 1 && wl == Low )     [i:S].east.acquire ->
             ascend   -> [i].east.depart -> SCHEDULER[we-1][  ww][East][wl]
    | when ( we >= 1 && td == East && wl == High ) [i:S].east.acquire ->
             resetlow -> [i].east.depart -> SCHEDULER[we-1][  ww][West][Low]
    | when ( ww == 0 && we >= 1 && wl == High )    [i:S].east.acquire ->
             resetlow -> [i].east.depart -> SCHEDULER[we-1][  ww][East][Low]

  // West-bound assignments to descend the lock system.
```

Lockset Behavior



Model Checking





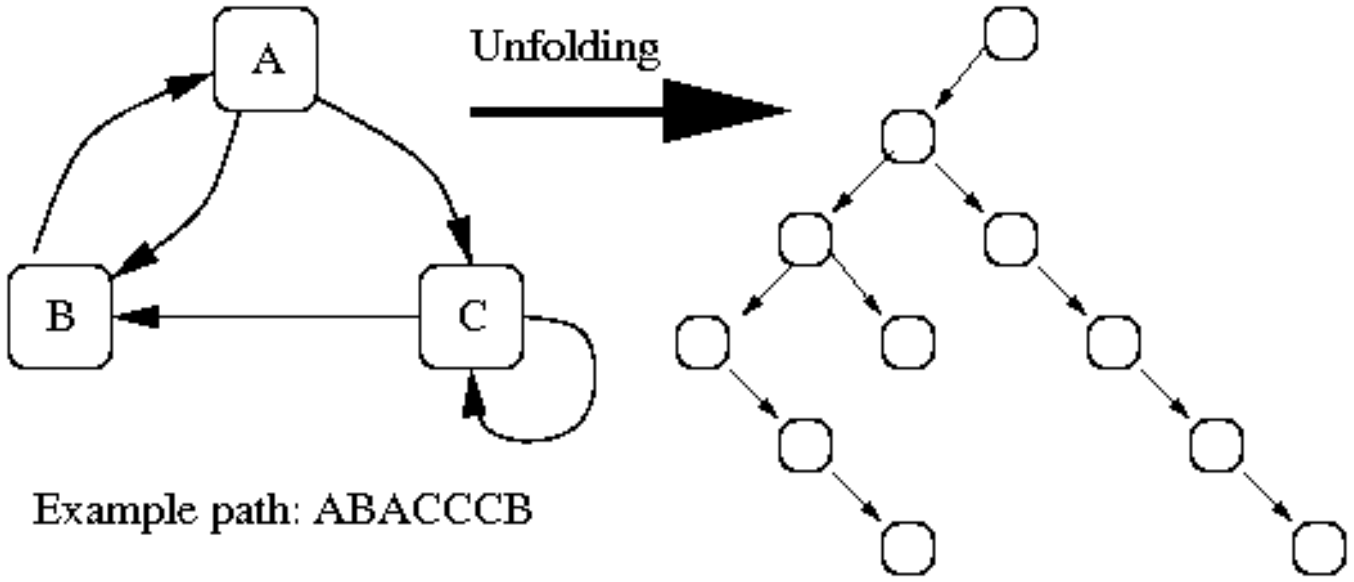
Design Properties

We would like to design systems that have properties that are guaranteed to be satisfied.

Safety	A safety property asserts that nothing bad happens..
Liveliness	A liveliness property asserts some good "eventually" happens..
Progress	A progress property asserts that it is always the case that eventually an "action" will be executed..

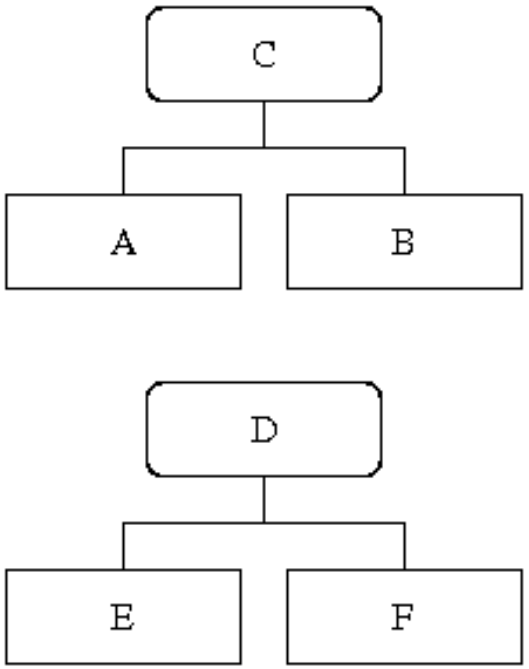
Model Checking

In practice the model checking procedure has two steps: (1) unfold the finite state machines into trees, and (2) exhaustively search the tree to see if the property specification is violated.

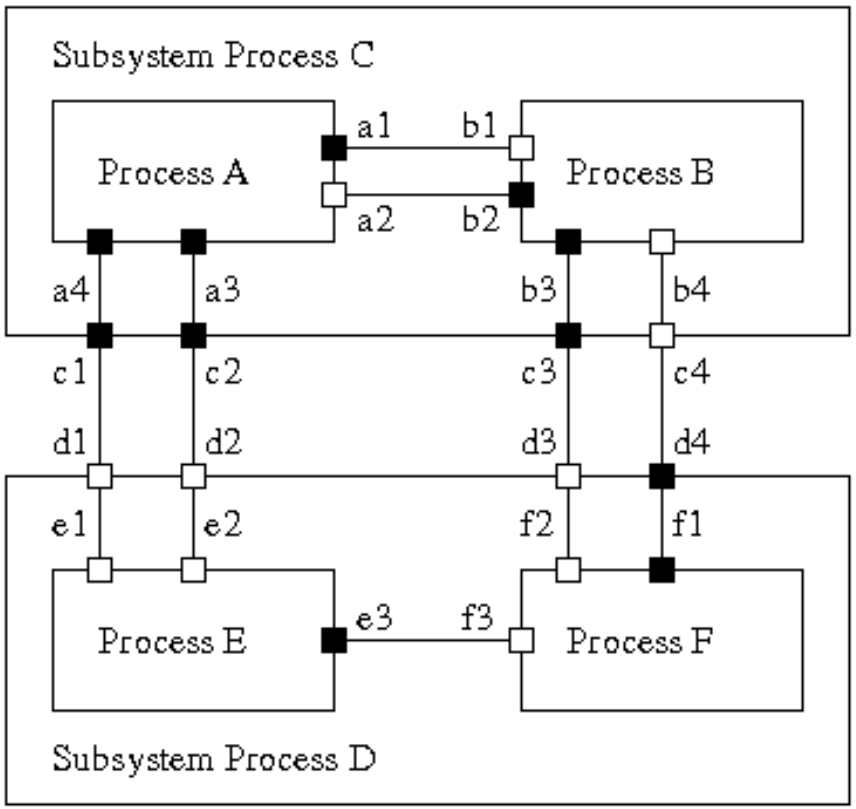


Model Checking

Subsystem–level Process Hierarchies

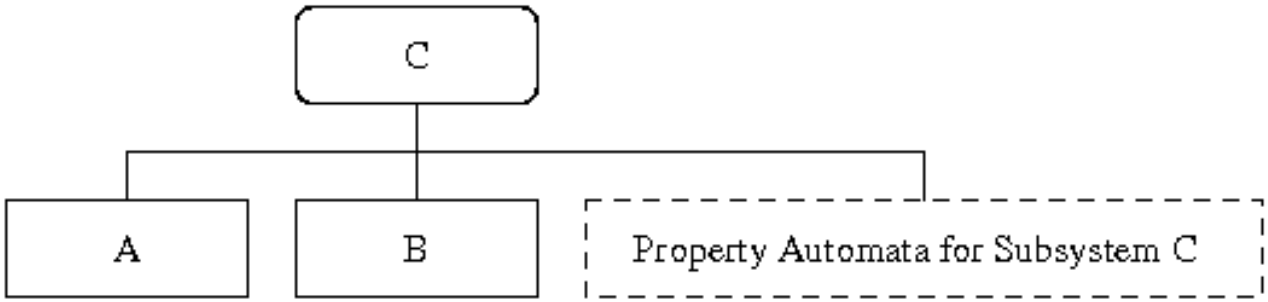


Plan View of Networked Processes

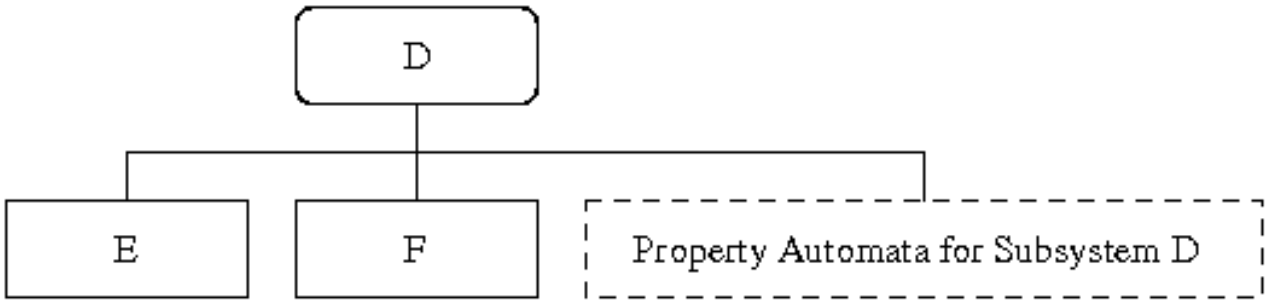


Sequence of Simplified Process Hierarchies

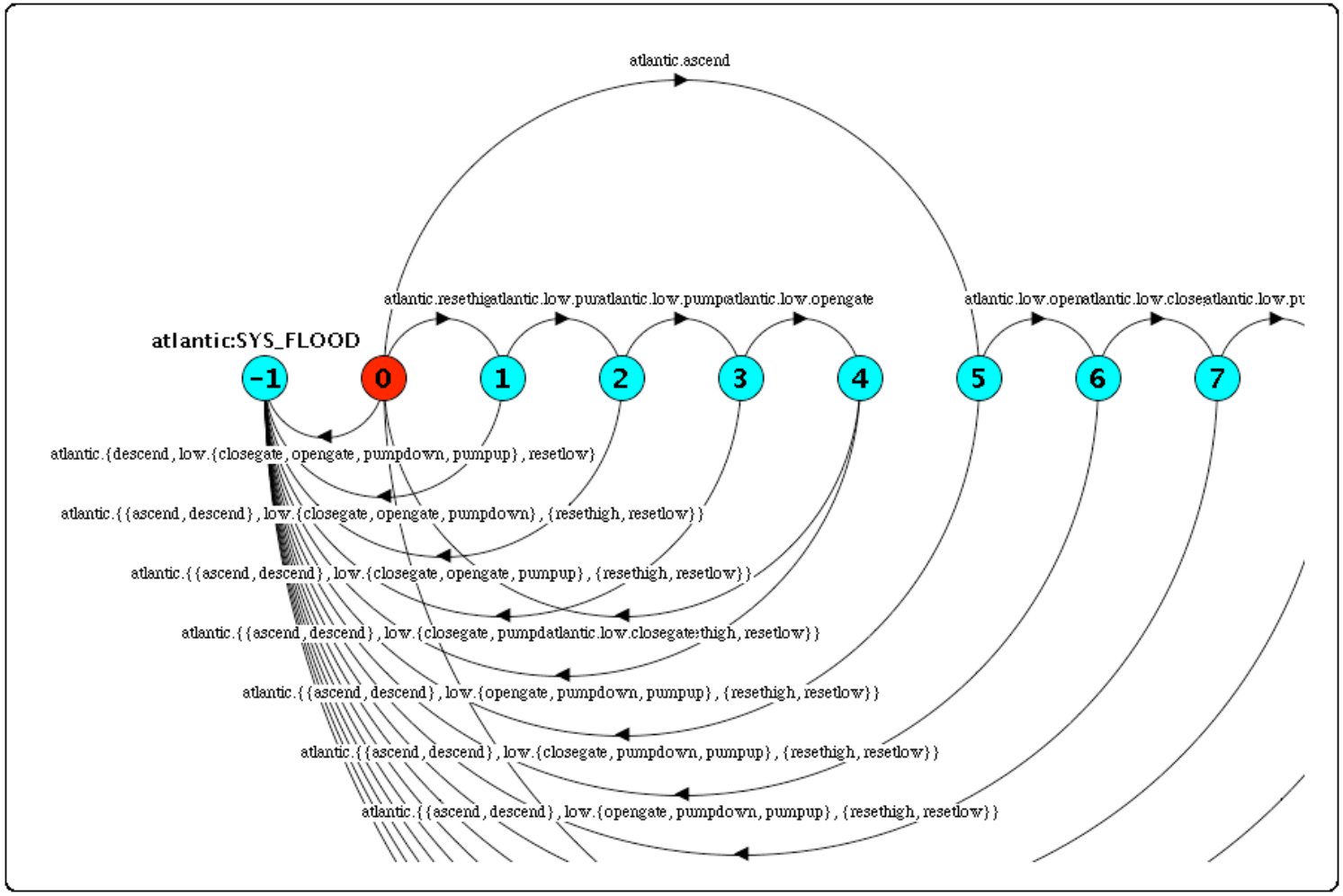
Problem 1. Validate Behavior of Subsystem C



Problem 2. Validate Behavior of Subsystem D



System Flood Property



Conclusions

This is a work in progress -- so what's next?

- Models for System-Level Operations (I.e., the Full Canal)
- Sensors, Non-Deterministic Models of Travel Demand
- Use of abstraction to simplify complexity of validation computations