A Visual Formalism for Interacting Systems

Model-Based Testing Workshop
London, 19 April 2015

Paul C. Jorgensen

Outline

- Introduction and goals
- Mainline existing models
- Statecharts and Swim Lane Event-Driven Petri Nets
- Communication primitives
  - standard Petri Net patterns
  - ESML prompts
  - service requests
- Example: a garage door controller
  - Statechart model
  - Swim Lane Event-Driven Petri Net
- Conclusions
- Questions

Characteristics of Interacting Systems

- Systems are/can be independent
- Possibility of centralized control
- Systems (constituents) can “reside” in separate devices (true concurrency)
- Systems of Systems are an excellent example
- Models need to express...
  - device residence
  - communication among constituent systems
  - parallel execution
  - autonomy of constituents

Testing Interacting Systems

- Echoes normal software integration testing
  - units are well-tested, and deemed ready for integration
  - integration testing focuses in interfaces among well-tested units
- Issues for model-based testing
  - focus must be on communication among constituent systems
  - frequently event-driven
  - need to support composition (bottom-up development)
  - must support concurrency
- Candidate models
  - Specification and Development Language (SDL)
  - Statecharts
  - Swim Lane Event-Driven Petri Nets

Specification and Development Language (SDL)

- Hierarchical block structure
  - blocks are/can be system constituents
  - blocks can be decomposed (as with dataflow diagrams)
- Communication via channels
- Behavior represented with finite state machines
- Very similar to flowcharts
- Fair commercial tool support

SDL Limitations

- Behavior models limited to constituent level
- Composing constituent level finite state machines quickly leads to the “finite state machine explosion”
  - where are dependencies expressed?
  - one finite state machine model of an elevator system with 3 lifts and nine floors has 110K+ states!
- Not a viable candidate.

Statecharts

- Powerful
  - very expressive, but can be very complex
  - Difficult to fully comprehend/review
- Hierarchical
  - deals effectively with size
  - Elevator statechart fits on 4 pages
- Concurrent regions are perfect for constituents
  - communication by broadcasting mechanism
- Executable
  - requires an engine
  - automatic generation of test scenarios (by customer!)
- Can only be composed in very limited circumstances
- Works best as a top-down strategy

Event-Driven Petri Nets (EDPNs)

- Extension to basic Petri nets
- Specific treatment of events
- Can show event quiescence
- EDPN elements
  - (P, D, S, In, Out)
  - P is a set of port events
  - D is a set of data places
  - S is a set of transitions
  - $In \subseteq (P \cup D) \times S$
  - $Out \subseteq S \times (P \cup D)$

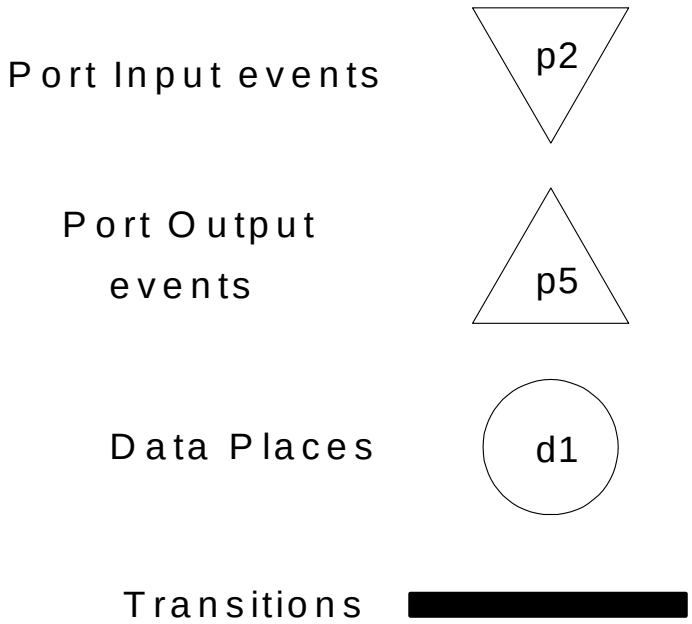


Figure 1 Event-Driven Petri Net elements

Swim Lane Event-Driven Petri Nets

- Formally equivalent to UML types I, II, and III statecharts
- Swim lanes correspond to constituents
- Petri net basis can support an engine
- Very expressive, Petri net based communication among constituents
- Easily composed
- Biggest disadvantage? Diagrams are very space consuming; don't scale up well
  - can resolve this with mapping Swim Lane EDPNs into a database, and composing distinct database populations.

Example: The Garage Door Controller

- Components:
  - drive motor
  - door tracks with sensors
  - control device (wireless keypad)
  - light beam sensor
  - obstacle sensor

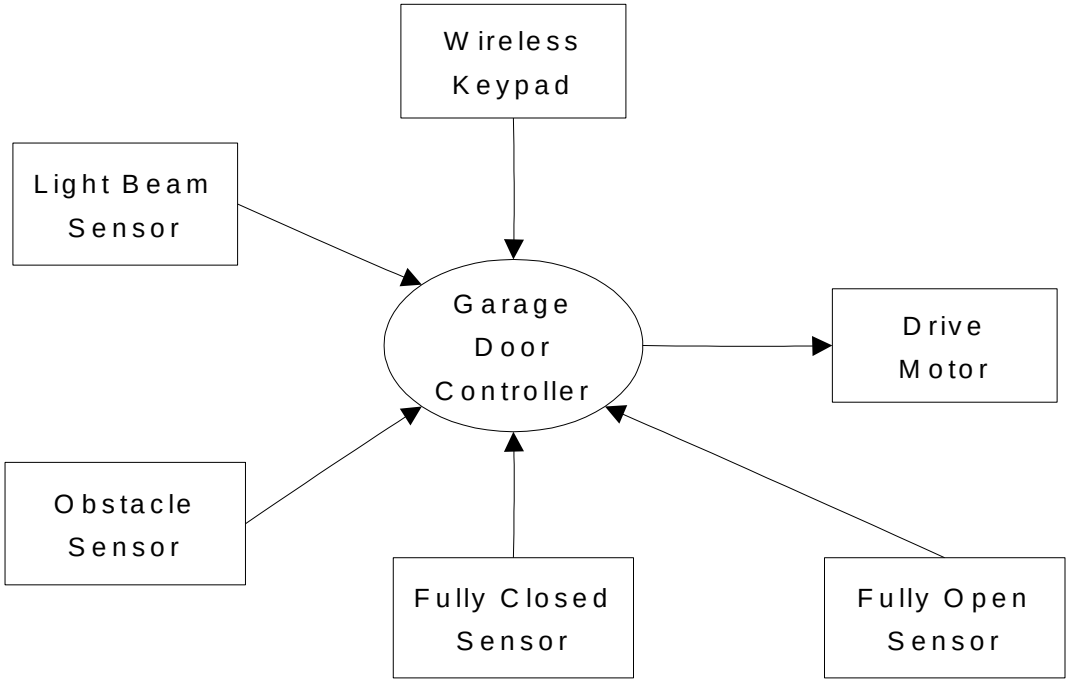


Figure 10 SysML Context Diagram of the Garage Door Controller

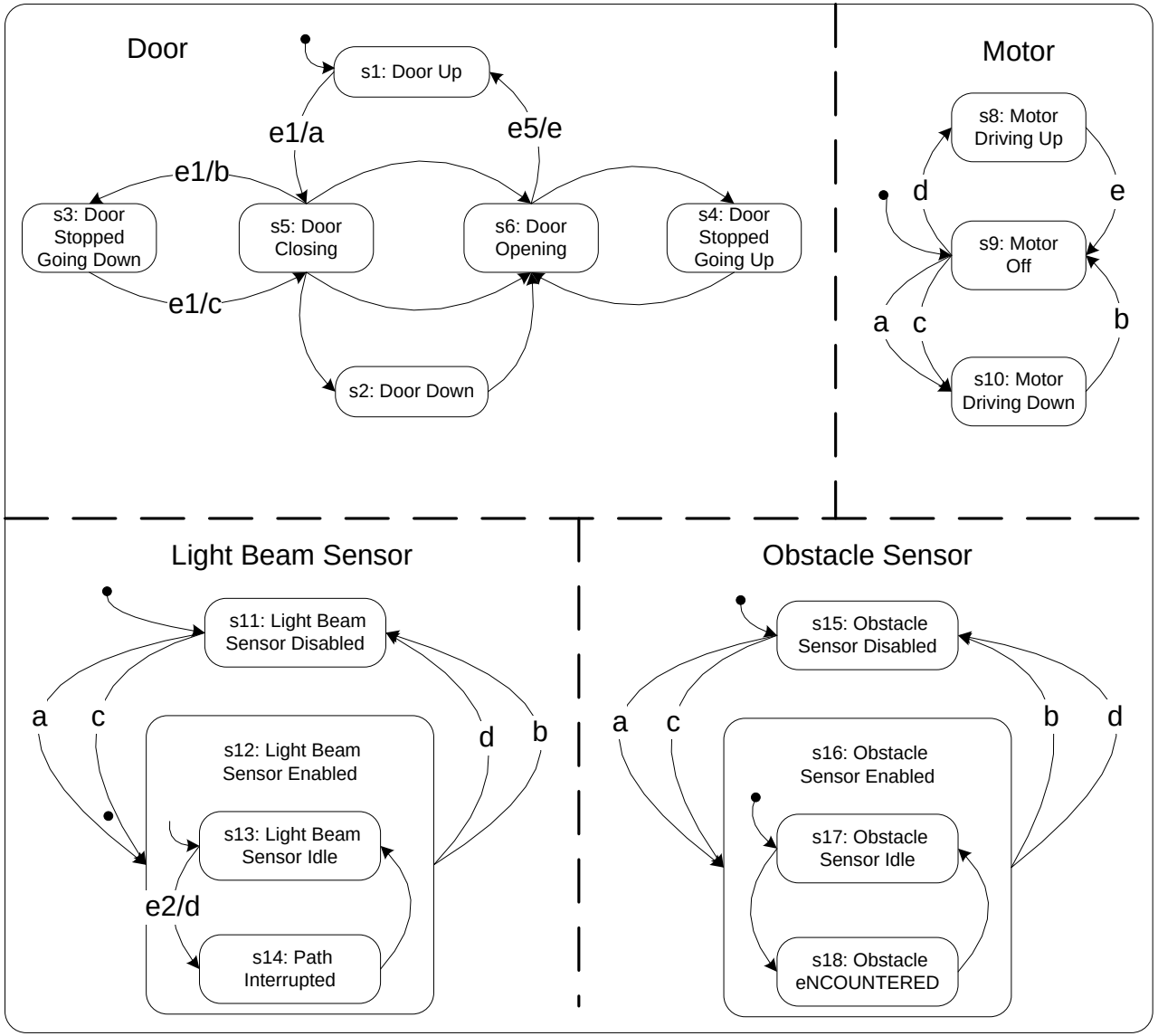
Sample Garage Door Controller Behaviors

- When the door is open, a signal from the wireless keypad triggers the drive motor.
- When the door is in motion, either opening or closing, a signal from the wireless keypad triggers the motor to stop.
- When the door is closing, the Light Beam and the Obstacle Sensors are enabled.
- Stopping a closing door disables the Light Beam and the Obstacle Sensors.
- An input from either the Light Beam or the Obstacle Sensor triggers the drive motor to stop and then begin opening.

Statechart Model of the Garage Door Controller

- Input events
  - e1: wireless keypad signal
  - e2: light beam interruption sensed
  - e3: obstacle sensed
  - e4: end of down track reached
  - e5: end of up track reached
- Output actions
  - a1: start drive motor down
  - a2: start drive motor up
  - a3: stop drive motor

Statechart Model of the Garage Door Controller



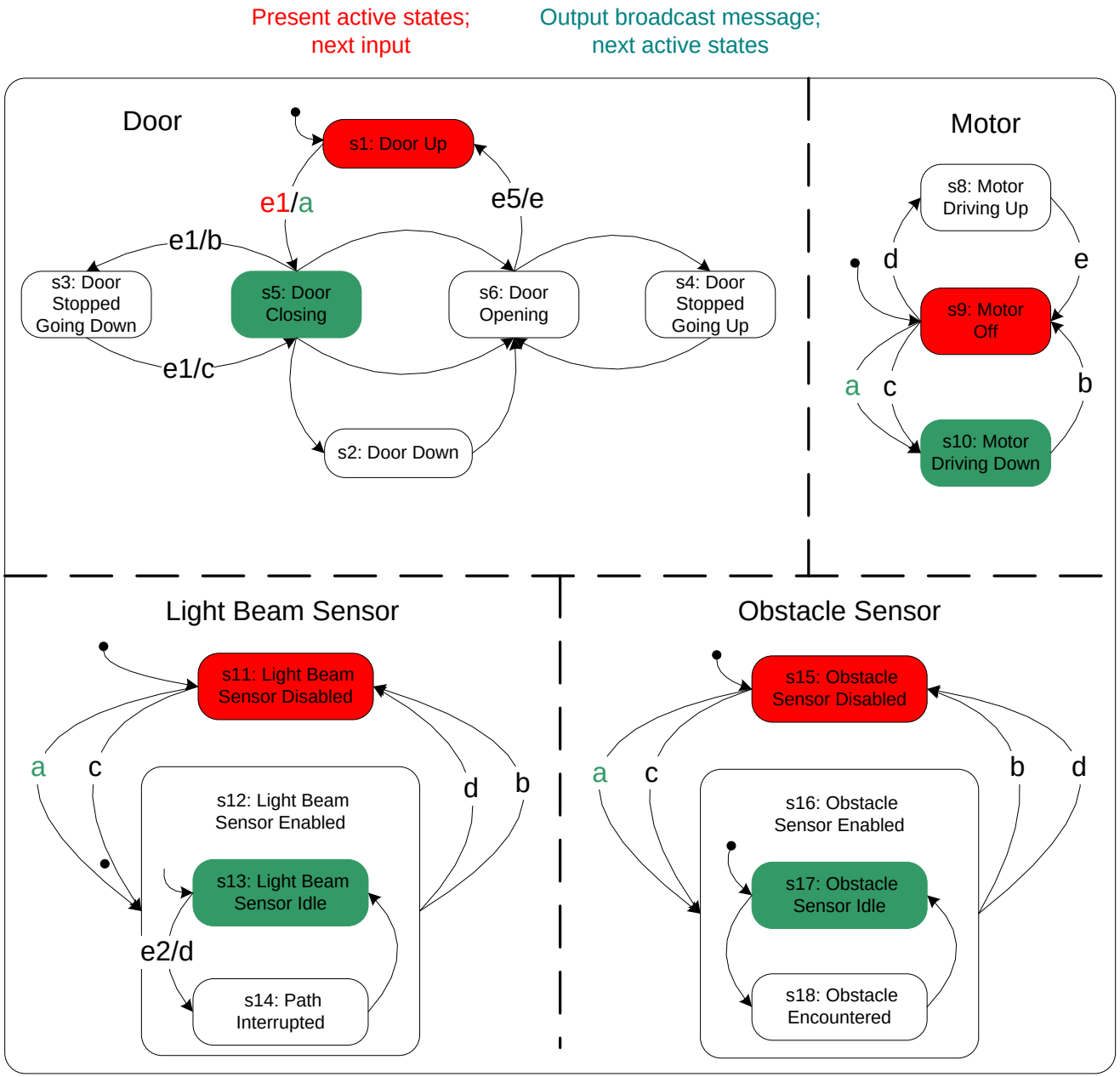
Sample Scenario

- Pre-condition: the door is up (open)
- Sequence of input events: <e1, e1, e1, e2, e5 >
- Post-condition: the door is up (open)
- Narrative description:
- Initially, the garage door is open. When a control signal occurs, the door begins to close. A second signal stops the door part way. A third signal resumes closing the door. A light beam interruption reverses the motor so the door is opening. When the end of up track sensor is reached, the motor stops and the door is fully open.

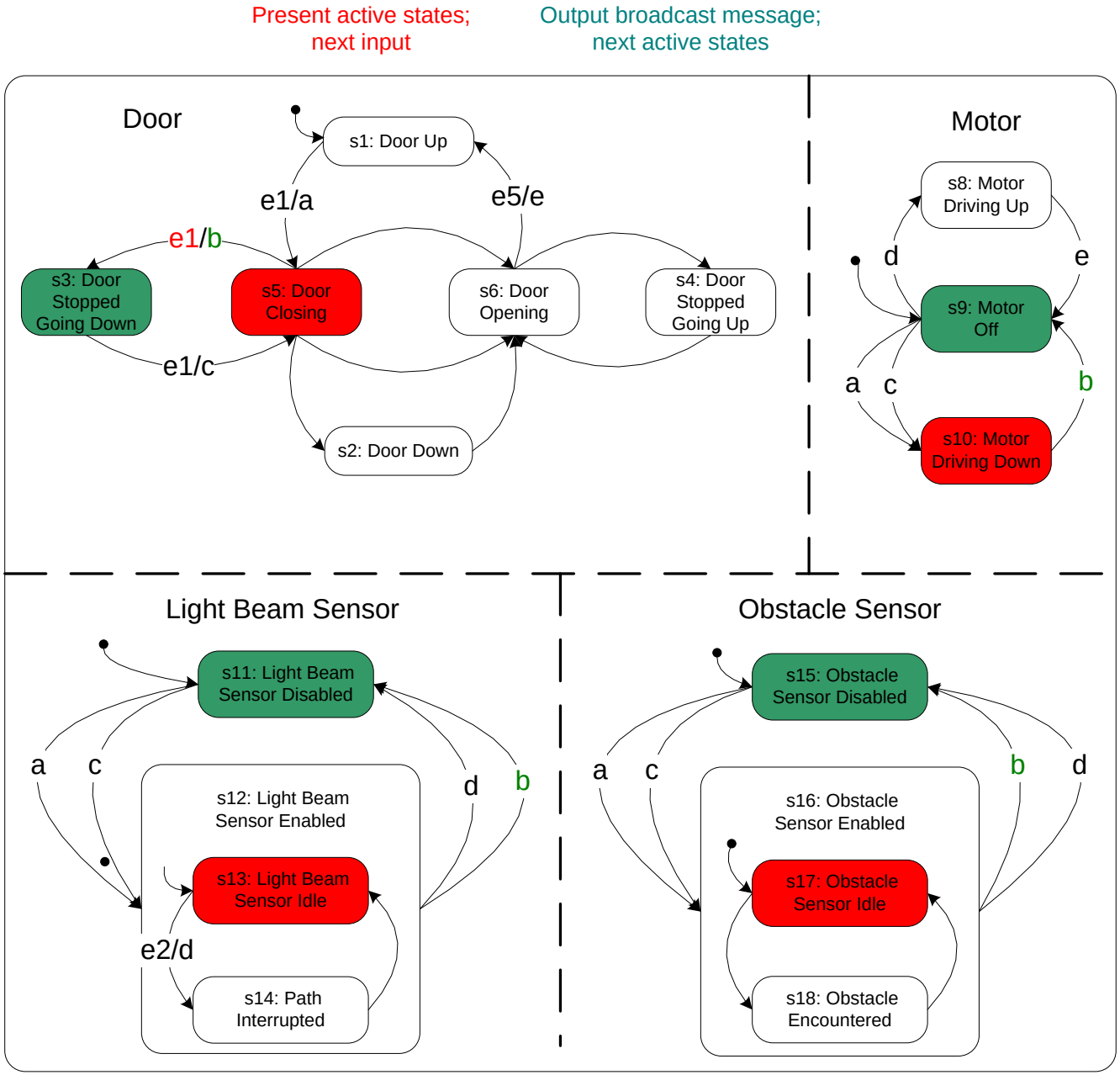
Statechart Execution Table

Garage Door Statechart Execution Table						
	In States					
Step	Door	Motor	Light Beam Sensor	Obstacle Sensor	Input Event	Broadcast Output
0	s1	s9	s11	s15	e1	a
1	s5	s10	s12, s13	s16, s17	e1	b
2	s3	s9	s11	s15	e1	c
3	s5	s10	s12, s13	s16, s17	e2	d
4	s6	s8	s12, s14	s15	e5	e
5	s1	s9	s12, s14	s15		

Step 1


Present active states;
next input
Output broadcast message;
next active states
Door
s1: Door Up
e1/a
e5/e
e1/b
s3: Door
Stopped
Going Down
s5: Door
Closing
s6: Door
Opening
s4: Door
Stopped
Going Up
e1/c
s2: Door Down
Motor
s8: Motor
Driving Up
d
e
s9: Motor
Off
a
c
b
s10: Motor
Driving Down
Light Beam Sensor
s11: Light Beam
Sensor Disabled
a
c
d
b
s12: Light Beam
Sensor Enabled
s13: Light Beam
Sensor Idle
e2/d
s14: Path
Interrupted
Obstacle Sensor
s15: Obstacle
Sensor Disabled
a
c
b
d
s16: Obstacle
Sensor Enabled
s17: Obstacle
Sensor Idle
s18: Obstacle
Encountered

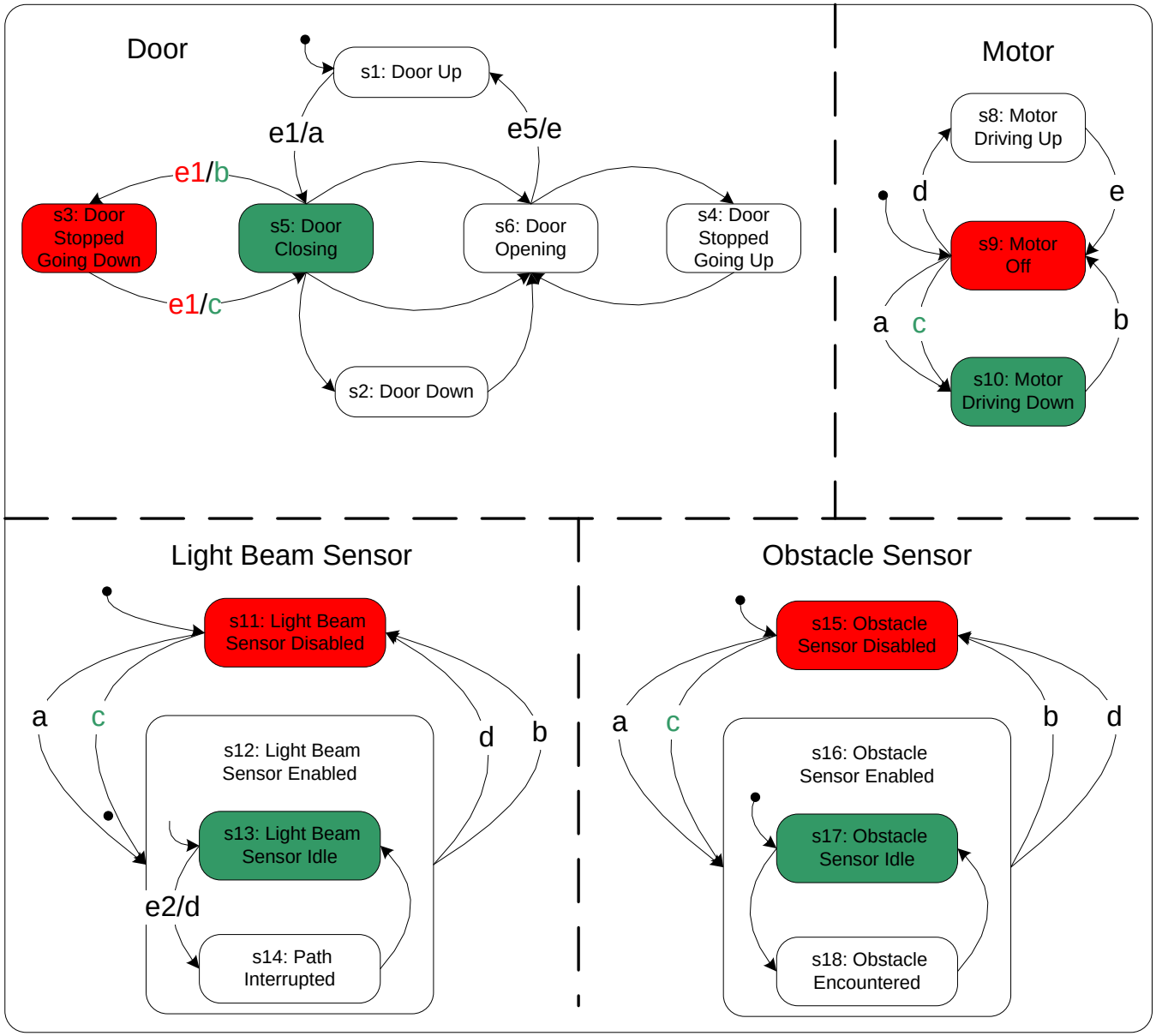
Step 2


Present active states;
next input
Output broadcast message;
next active states
Door
s1: Door Up
e1/a
e5/e
e1/b
s3: Door
Stopped
Going Down
s5: Door
Closing
s6: Door
Opening
s4: Door
Stopped
Going Up
e1/c
s2: Door Down
Motor
s8: Motor
Driving Up
d
e
s9: Motor
Off
a
c
b
s10: Motor
Driving Down
Light Beam Sensor
s11: Light Beam
Sensor Disabled
a
c
d
b
s12: Light Beam
Sensor Enabled
s13: Light Beam
Sensor Idle
e2/d
s14: Path
Interrupted
Obstacle Sensor
s15: Obstacle
Sensor Disabled
a
c
b
d
s16: Obstacle
Sensor Enabled
s17: Obstacle
Sensor Idle
s18: Obstacle
Encountered

Step 3

Present active states; next input

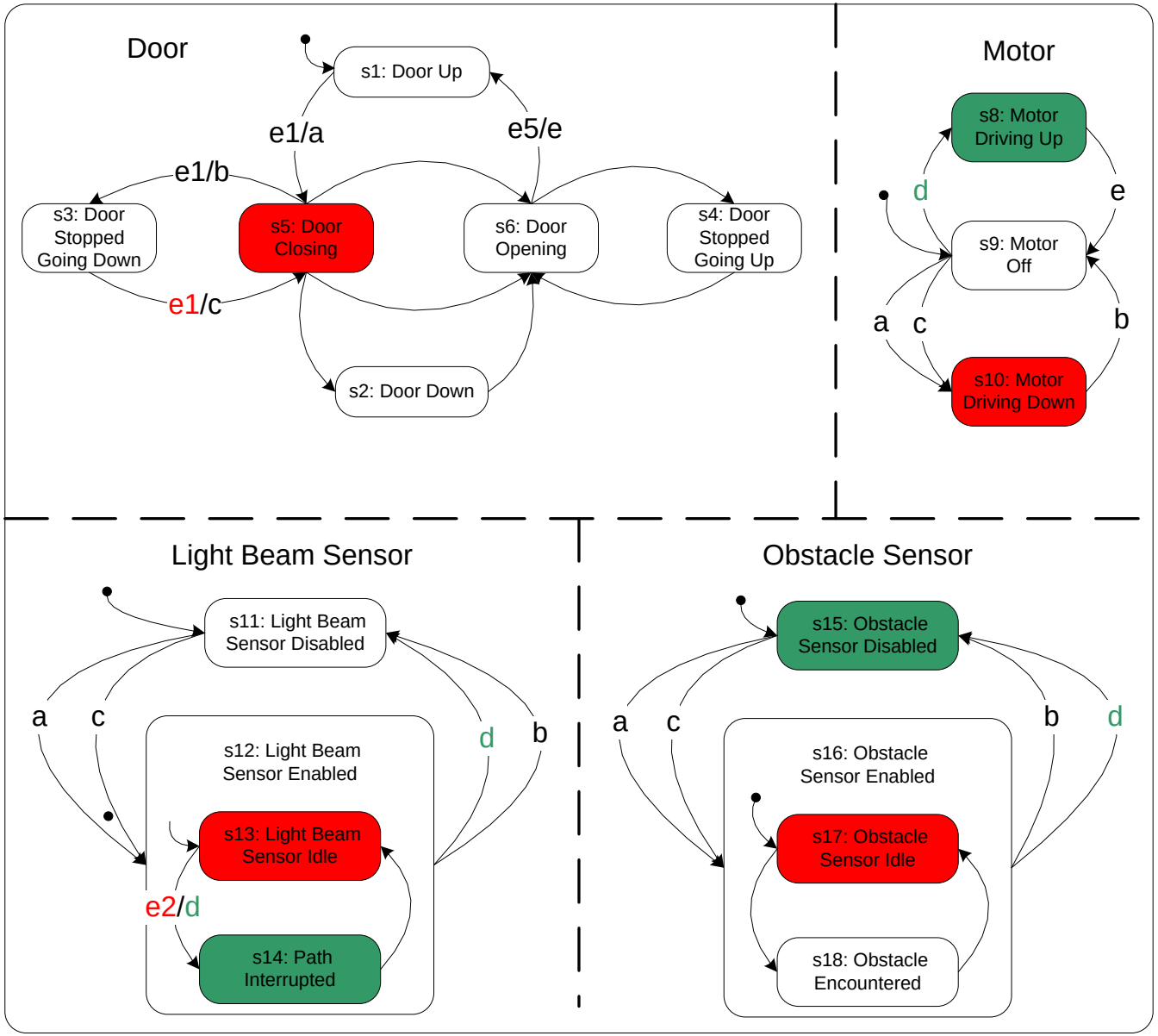
Output broadcast message; next active states



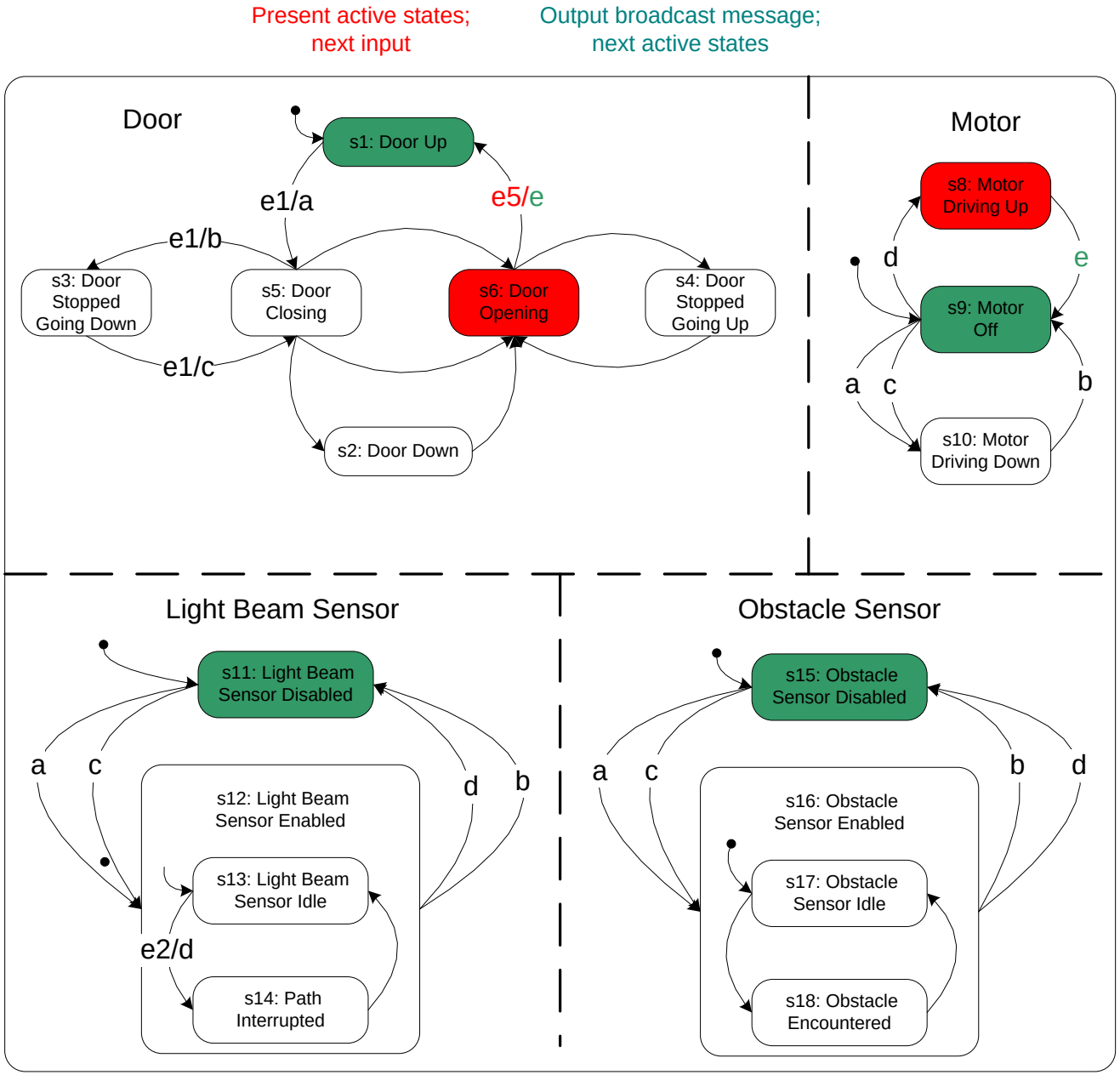
Present active states;
next input

Output broadcast message;
next active states

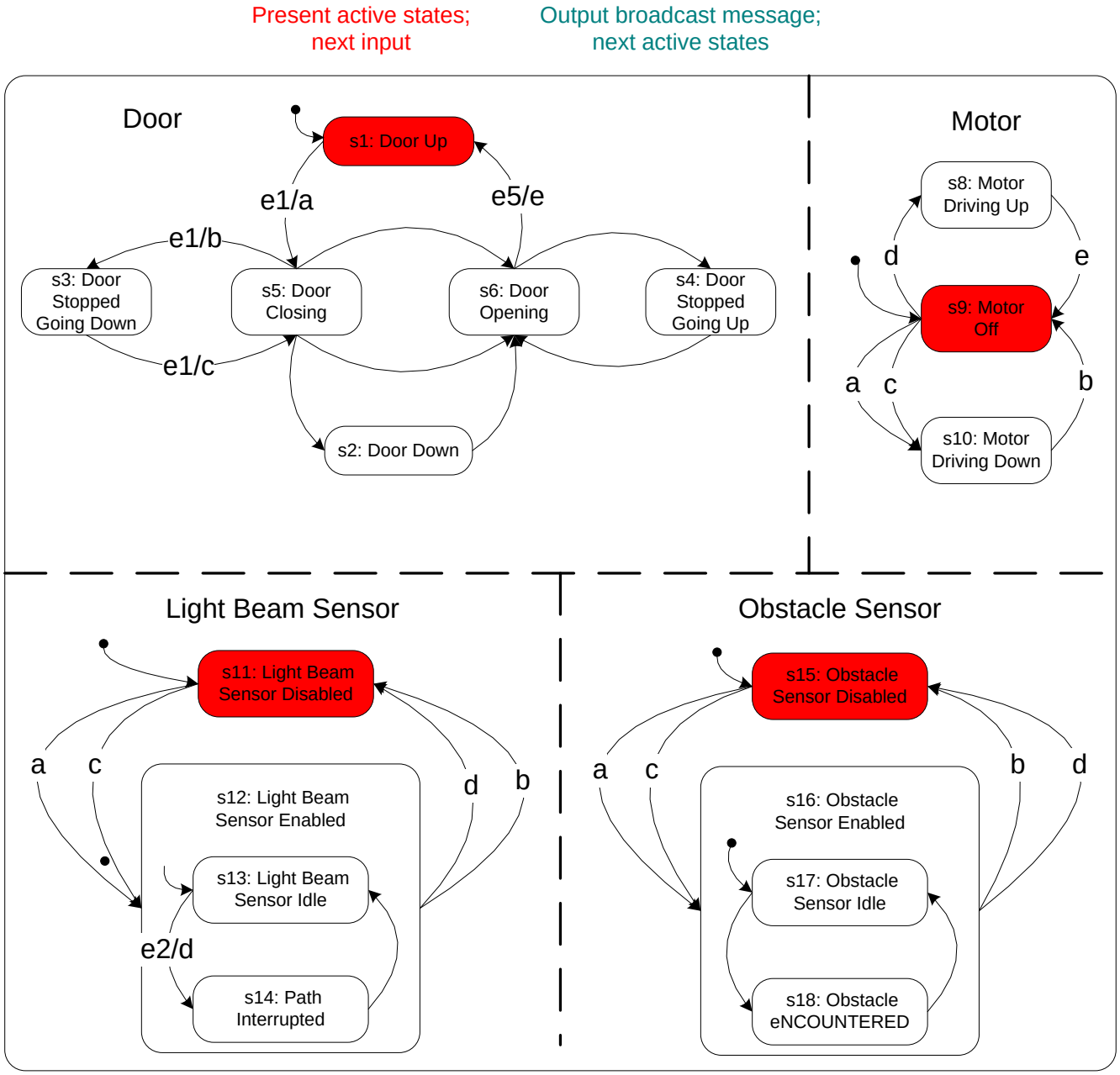
Step 4



Step 5


Present active states; next input
Output broadcast message; next active states
Door
s1: Door Up
e1/a
e5/e
e1/b
s3: Door Stopped Going Down
s5: Door Closing
s6: Door Opening
s4: Door Stopped Going Up
e1/c
s2: Door Down
Motor
s8: Motor Driving Up
d
e
s9: Motor Off
a
c
b
s10: Motor Driving Down
Light Beam Sensor
s11: Light Beam Sensor Disabled
a
c
d
b
s12: Light Beam Sensor Enabled
s13: Light Beam Sensor Idle
e2/d
s14: Path Interrupted
Obstacle Sensor
s15: Obstacle Sensor Disabled
a
c
b
d
s16: Obstacle Sensor Enabled
s17: Obstacle Sensor Idle
s18: Obstacle Encountered

Step 6

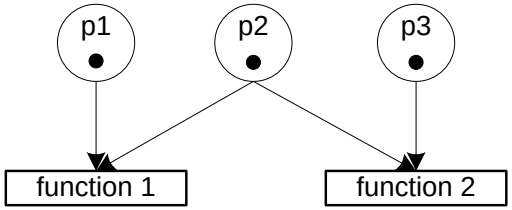

Present active states; next input
Output broadcast message; next active states
Door
s1: Door Up
e1/a
e5/e
e1/b
s3: Door Stopped Going Down
s5: Door Closing
s6: Door Opening
s4: Door Stopped Going Up
e1/c
s2: Door Down
Motor
s8: Motor Driving Up
d
e
s9: Motor Off
a
c
b
s10: Motor Driving Down
Light Beam Sensor
s11: Light Beam Sensor Disabled
a
c
d
b
s12: Light Beam Sensor Enabled
s13: Light Beam Sensor Idle
e2/d
s14: Path Interrupted
Obstacle Sensor
s15: Obstacle Sensor Disabled
a
c
b
d
s16: Obstacle Sensor Enabled
s17: Obstacle Sensor Idle
s18: Obstacle eNCOUNTERED

Communication Primitives (Patterns)

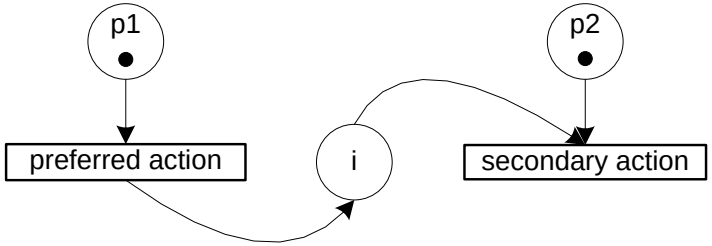
- Traditional Petri net patterns
- Extended Systems Modeling Language (ESML) prompts
- Service requests and responses

- These capture the ways in which interacting systems communicate.

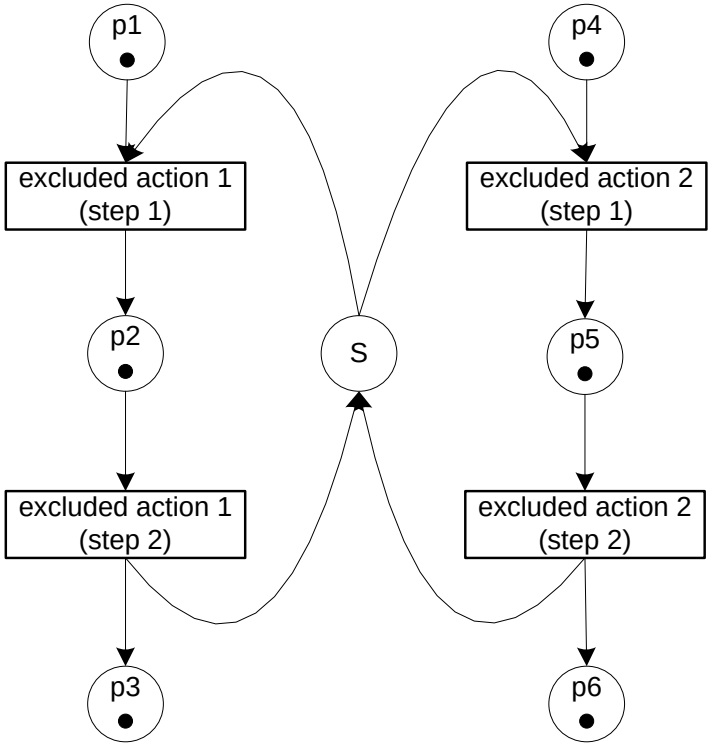
Traditional Petri Net Patterns



Conflict



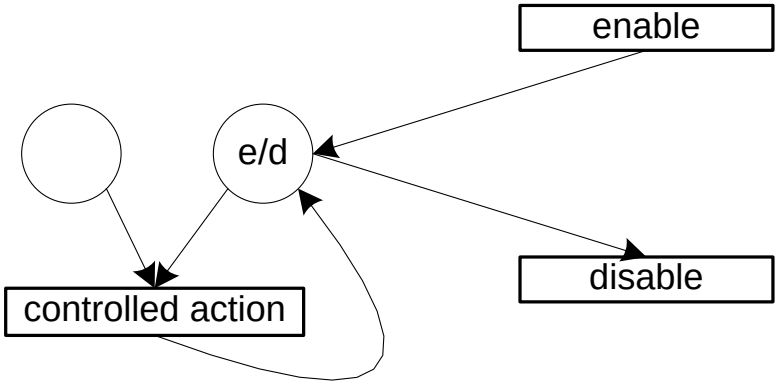
Interlock



Mutual Exclusion

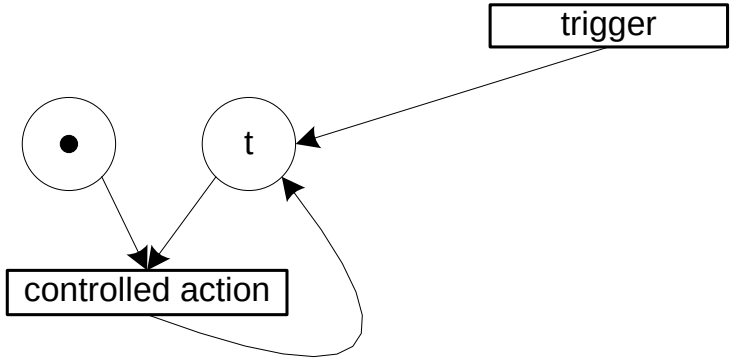
ESML Prompts: Enable, Disable, and Activate

- Place e/d must be marked before the controlled action can fire.
- Once enabled, the controlled action remains enabled until it is disabled.
- the Enable/Disable sequence is known as the Activate prompt.



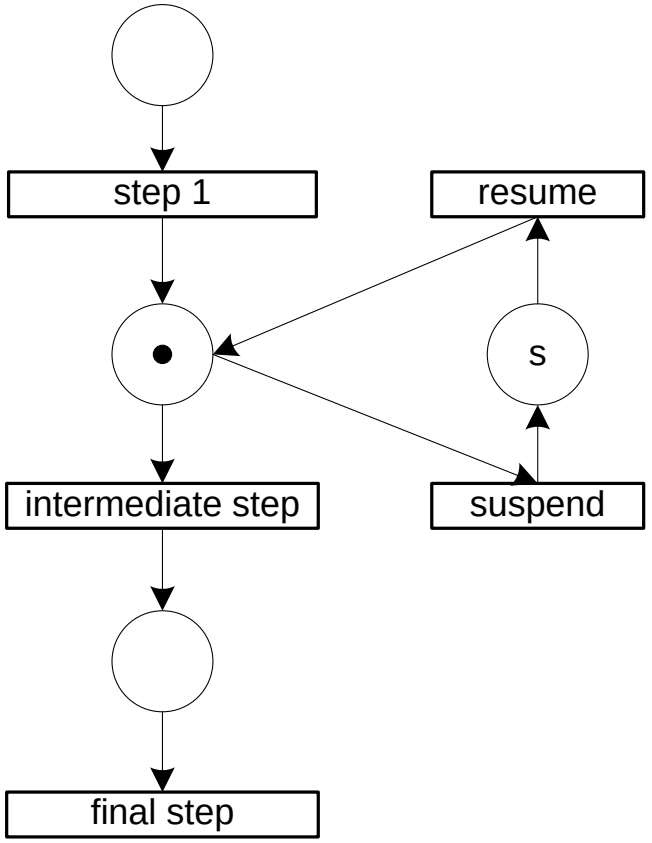
ESML Prompt: Trigger

- Once place t is marked, the controlled action fires immediately.
- Compare with Enable.
- Open Question: is Trigger a one-time prompt or does it repeat like Enable?
- Can use the output arrow to control this choice.



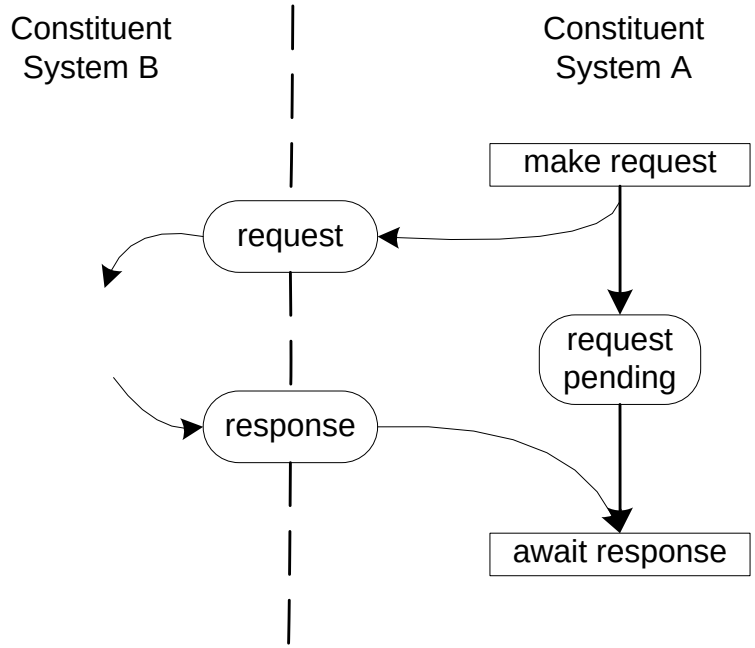
ESML Prompts: Suspend, Resume, and Pause

- Suspend and Resume are the reverse of Enable and Disable.
- Intent is to preserve intermediate results.
- Place s is an interlock that assures that a sequence must first be suspended before it can be resumed.



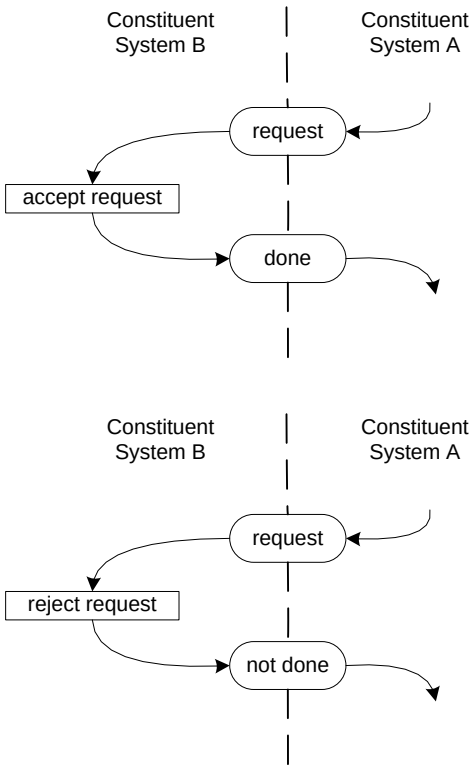
Service Requests and Responses

- System A requests a service from system B.
- The request pending place acts like an Enable.
- The response from system B acts like a Trigger.
- Systems A and B are in separate swim lanes (independent devices).

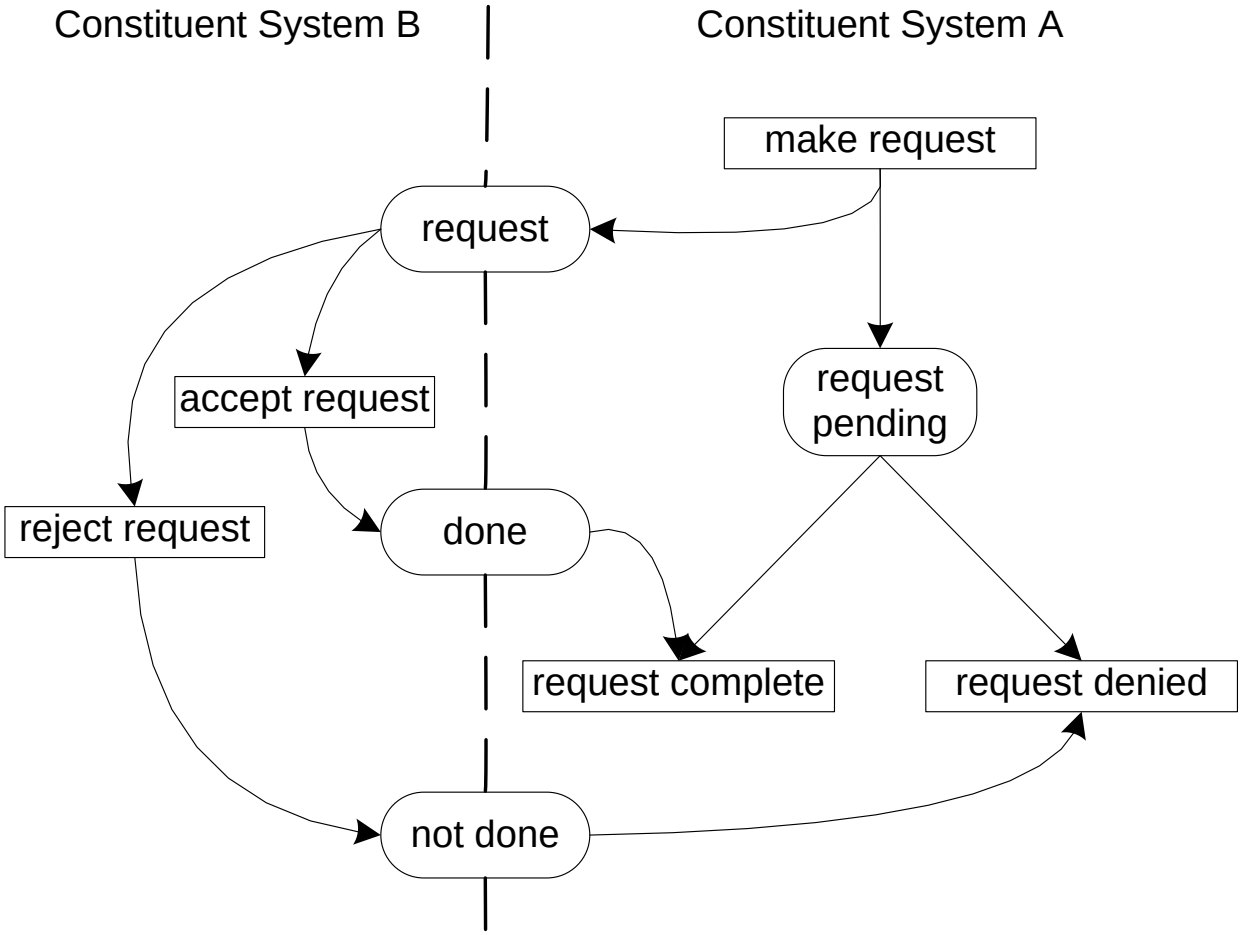


Accept and Reject Responses

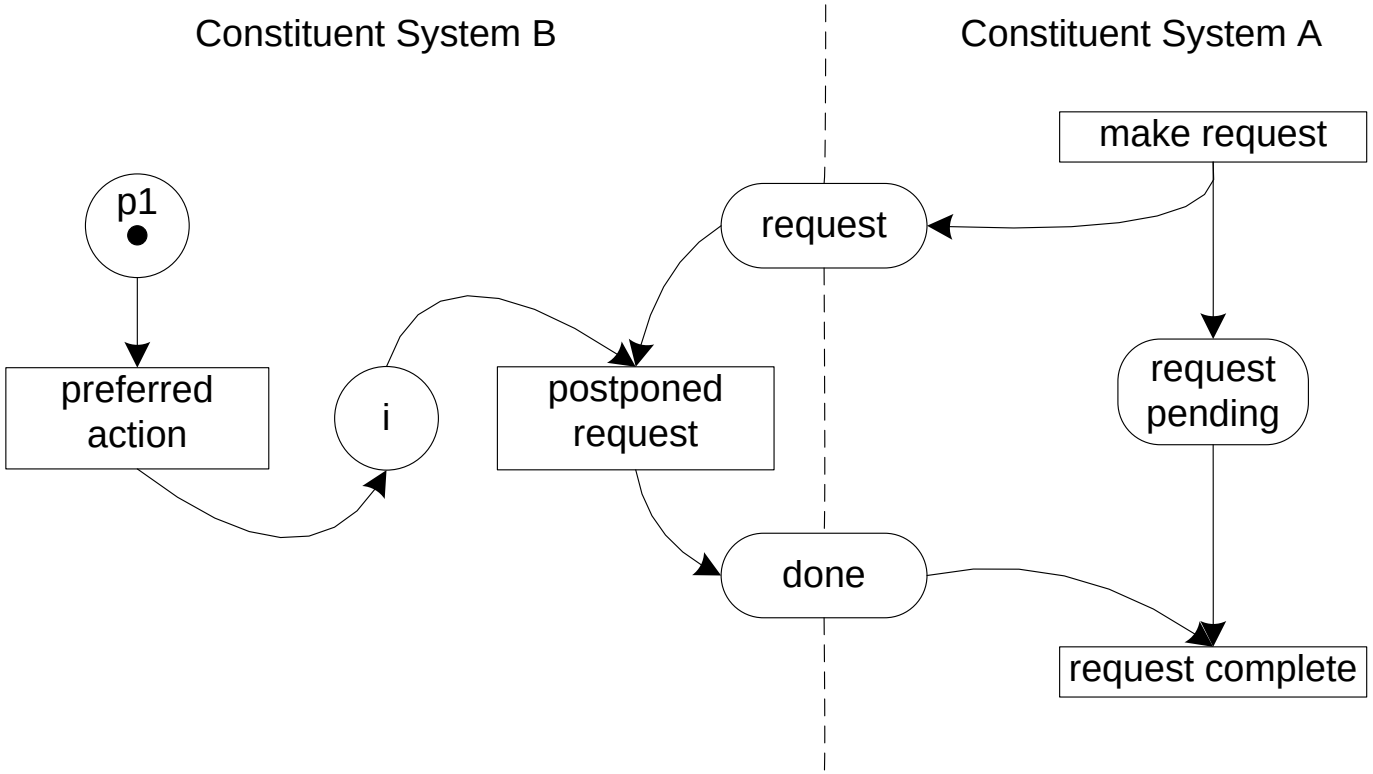
- Since systems A and B are independent, system B can either accept or reject the service request from system A.
- (More complete figure on the next slide)



Accept and Reject Responses



Postpone Response



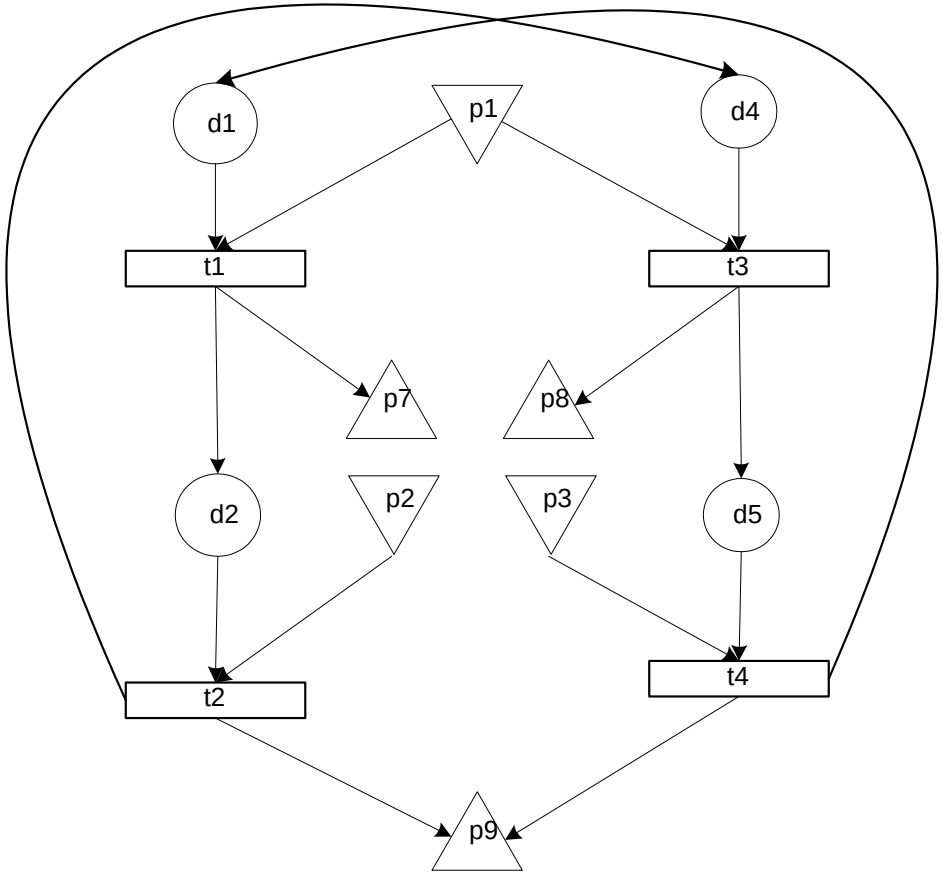
Note presence of Enable, Interlock, and Disable.

EDPN elements for the Garage Door Controller

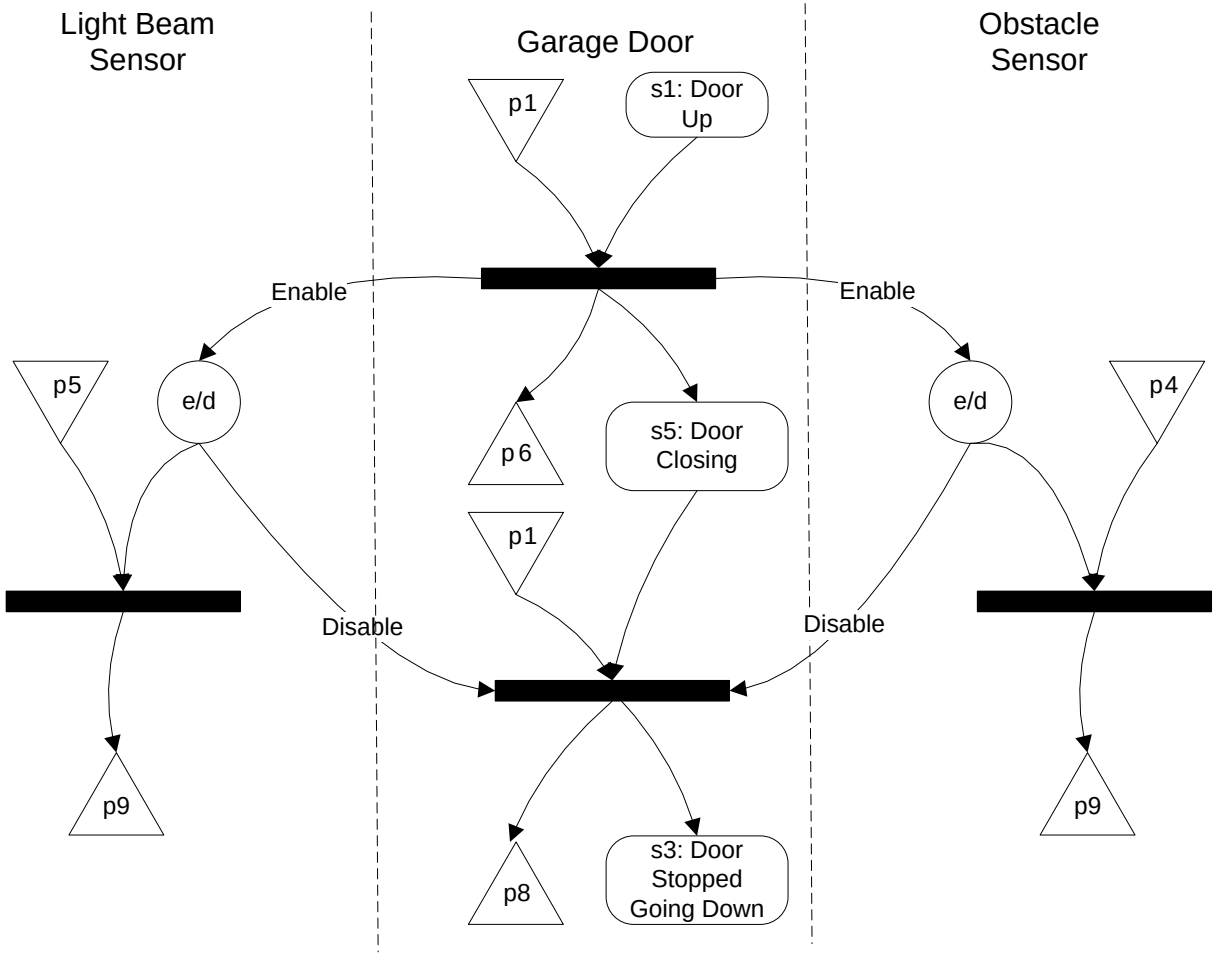
Input events	Output events (actions)	Data Places
p1: wireless keypad signal	p7: start drive motor down	d1: Door Up
p2: end of down track hit	p8: start drive motor up	d2: Door Closing
p3: end of up track hit	p9: stop drive motor	d3: Door Stopped going down
p4: brief motor pause		d4: Door Down
p5: light beam sensor		d5: Door Opening
p6: obstacle sensor		d6: Door Stopped going up

EDPN of the basic garage door operation

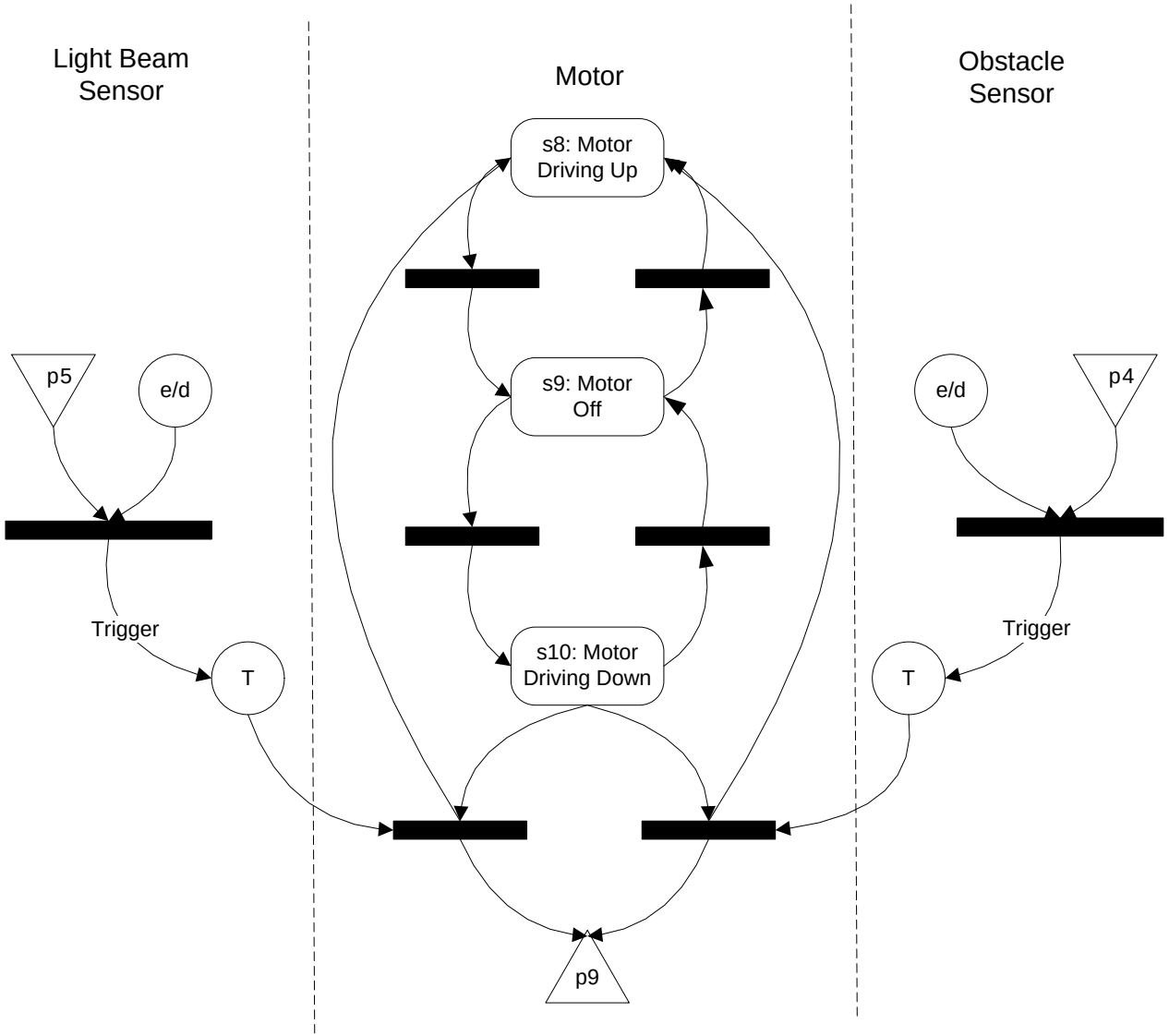
- Sample scenarios
  - Close an open door (transition sequence: <t1, t2>)
  - Open a closed door <t3, t4>
  - Open closed door and then close it. <t3, t4, t1, t2>



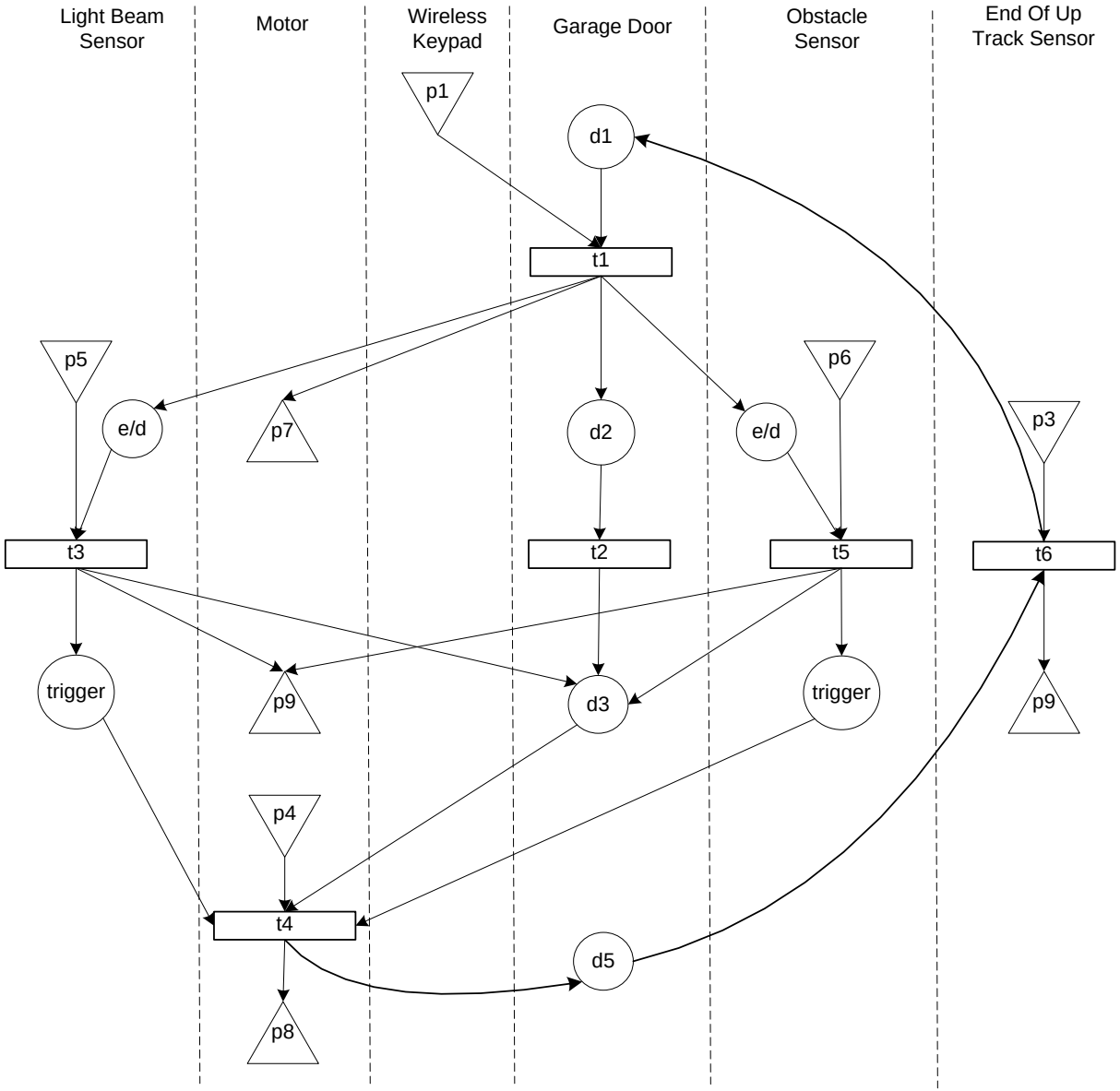
Swim Lane detailed view of safety feature enabling and disabling



Detailed view of safety features and drive motor



More comprehensive view...



Summary of Swim Lane EDPNs

- they are a bottom-up approach
- they can be easily composed
- they permit focused description of interactions among constituents
- they can be used for automatic derivation of system test cases
- they support a useful hierarchy of system test coverage metrics

- BUT...the drawings do not scale up well

Swim Lane EDPN composition in a database

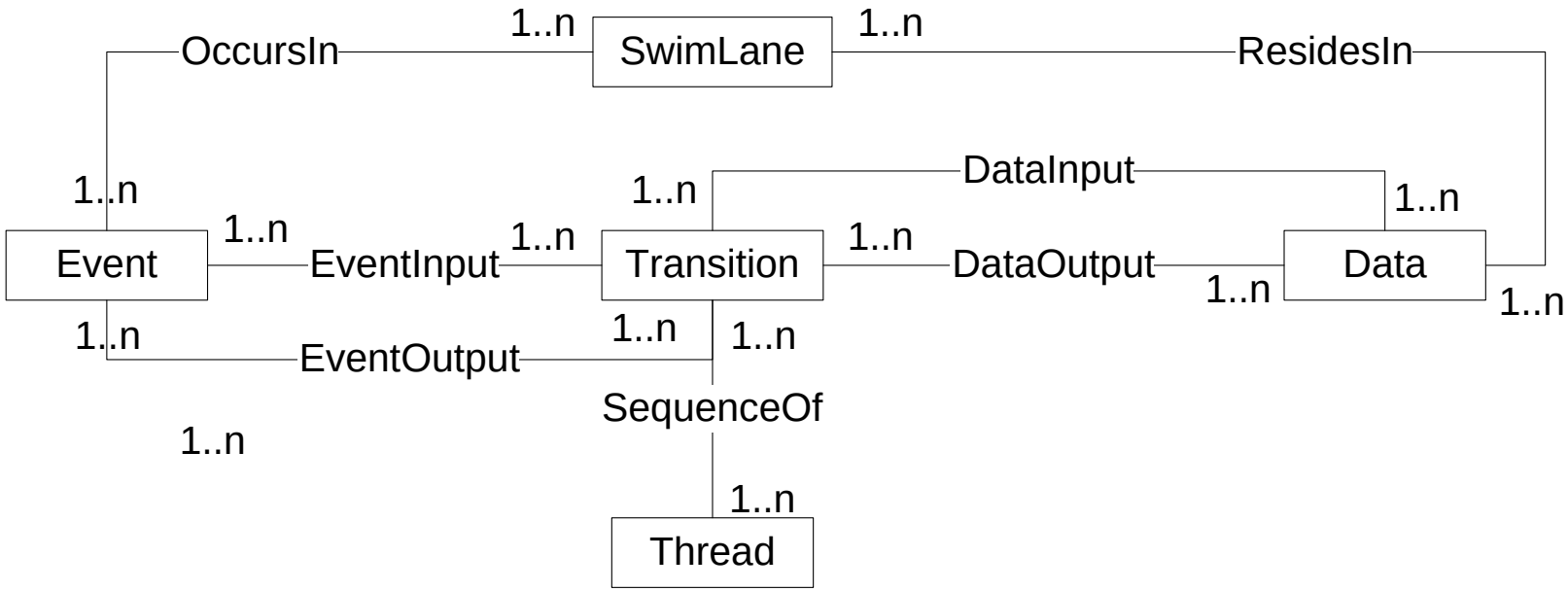
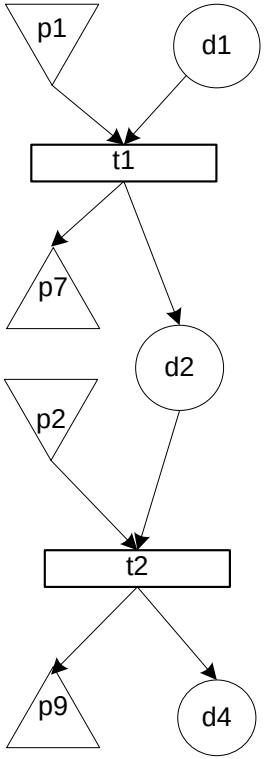


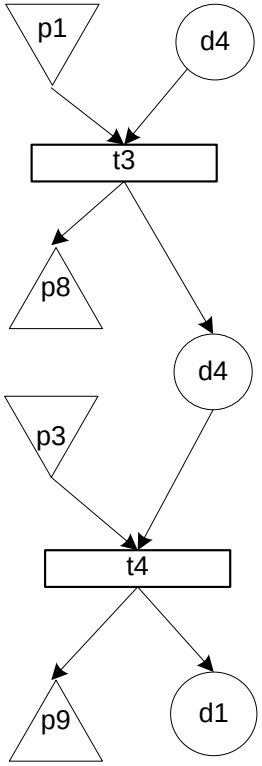
Figure 16 E/R Model of Swim Lane EDPN Database

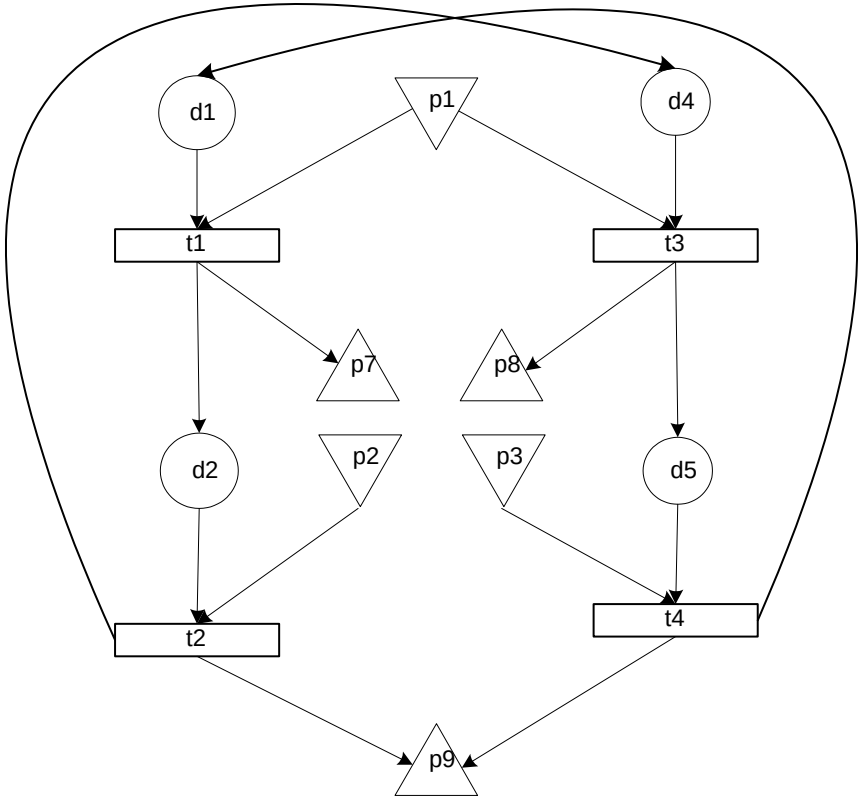
Composition Example

Door closing EDPN



Door Opening EDPN





Populated Databases

Database for Door Closing EDPN							
EventInput		EventOutput		DataInput		DataOutput	
Event	Transition	Event	Transition	Data	Transition	Data	Transition
p1	t1	P7	t1	d1	t1	d2	t1
p2	t2	P9	t2	d2	t2	D4	t2

Database for Door Opening EDPN							
EventInput		EventOutput		DataInput		DataOutput	
Event	Transition	Event	Transition	Data	Transition	Data	Transition
p1	t3	p7	t3	d3	t3	d4	t3
p3	t4	p8	t4	d4	t4	d1	t4

Database for Composition of Door Closing and Door Opening EDPNs							
EventInput		EventOutput		DataInput		DataOutput	
Event	Transition	Event	Transition	Data	Transition	Data	Transition
p1	t1	p6	t1	d1	t1	d2	t1
p1	t3	p7	t3	d2	t2	d3	t2
p2	t2	p8	t2	d3	t3	d4	t3
p3	t4	p8	t4	d4	t4	d1	t4

EDPN Test Coverage Metrics

Test Cover	Description
Ct	every transition
Cp	every data place
Cie	every input event
Coe	every output event
Ccontext	Cie in every context

Questions?

Thank You

Why these colors?