Modeling Software Behavior

Towhid Chowdhury

Introduction

- A software engineer needs to know how a software will behave
- Even in code-n-fix projects, developers are creating a mental behavior model of the software
- How do you visualize software behavior?

Behavior Modeling

- A behavior model needs to be representative of the problem
- It needs to cover nuances of the events and actions that will take place
- It needs to be good enough, but not overly complex, for the problem at hand
- What issues might a behavior model need to represent?

Behavioral Issues

- Sequence
  - Step by step execution.
  - Model represents each step in order.
- Selection
  - Multiple decisions and their results are represented in the model.
- Repetition
  - Model is able to represent repeated events while maintaining the previous state.

- Enable
  - When activity A enables activity B, activity B may execute but not immediately.
- Disable
  - When activity A disables activity B, activity B may not execute. [Immediate]
- Trigger
  - When activity A triggers activity B, B immediately executes.
- Activate
  - An enable/disable sequence performed by activity A for/on activity B.

Behavioral Issues (cont.d)

- Suspend
  - When A suspends B, B no longer executes but intermediate work is saved.
- Resume
  - When A resumes, B (which has been suspended) starts with the partially completed work that was in progress when it was suspended.
- Pause
  - A suspend/resume performed by A for/on B.

- Mutual Exclusion
  - Decisions are mutually exclusive.
- Concurrent execution
  - Multiple tasks are executed in parallel.
- Multiple-context output events
  - Depending on several previous tasks, the output of the current task changes.
- Asynchronous events
  - Tasks are executed out of order
- Event quiescence
  - Tasks are dependent on conditions.
- Context-sensitive input events
  - Same as multiple-context output events.

Behavior Models

1. Flowchart
2. Decision Tables
3. Finite State Machines
4. Petri Nets
5. Event Driven Petri Nets
6. Statecharts

Case Study - Smart Fuel Pump

- The system combines computer technology with traditional gas station equipment.
- Supports fuel-pumping devices with 2 interfaces:
  - One for the customer
  - One for the gas station attendant
- What behaviors do you need to model?

Smart Fuel Pump - Behavioral Issues

- Sequence
  - Transaction approval before fuel selection
- Selection
  - Fuel grade selection
- Repetition
  - Trigger squeeze and release
- Enable
  - Transaction approval enables pump
- Disable:
  - Transaction rejection disables pump
- Trigger:
  - Trigger squeeze triggers display updates
- Activate:
  - Nozzle removal activates trigger
- Suspend:
  - Trigger release during pumping
- Resume:
  - Trigger squeeze after a trigger release
- Pause:
  - Trigger squeeze and release; display resumes where it left off
- Mutual exclusion:
  - Fuel type selection

and many more.....

Smart Fuel Pump

- Will be used as the primary case study

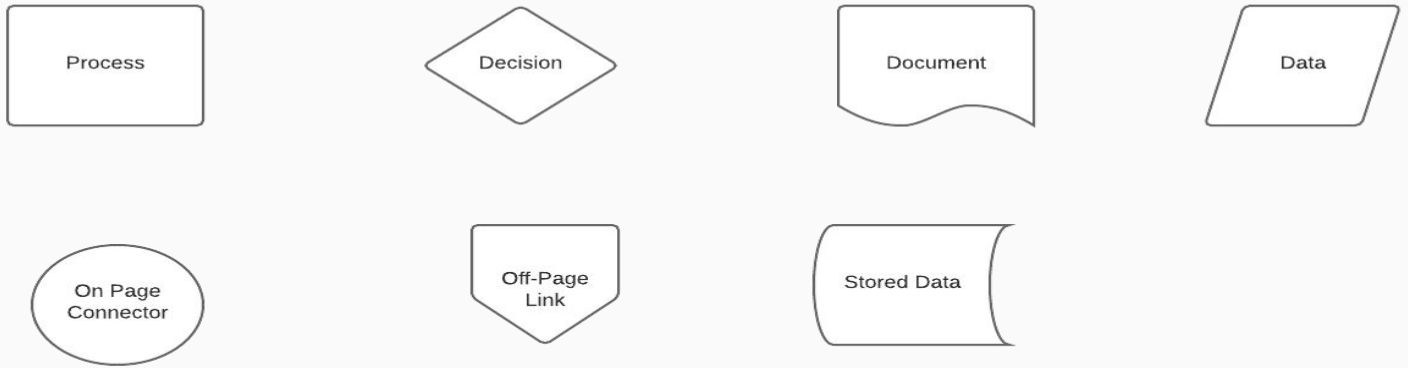
- Keep an eye out for the differences between various models for the Smart Fuel Pump.

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets	Event Driven Petri Nets	Statechart
Sequence	yes	no	yes	yes	yes	yes
Selection	yes	yes	yes	yes	yes	yes
Repetition	yes	yes	yes	yes	yes	yes
Enable	no	no	no	yes	yes	yes
Disable	no	no	no	yes	yes	yes
Trigger	no	no	no	yes	yes	yes
Activate	no	no	no	yes	yes	yes
Suspend	no	no	no	yes	yes	yes
Resume	no	no	no	yes	yes	yes
Pause	no	no	no	yes	yes	yes
Conflict	no	no	no	yes	yes	yes
Priority	no	no	no	yes	yes	yes

Flowcharts

Overview

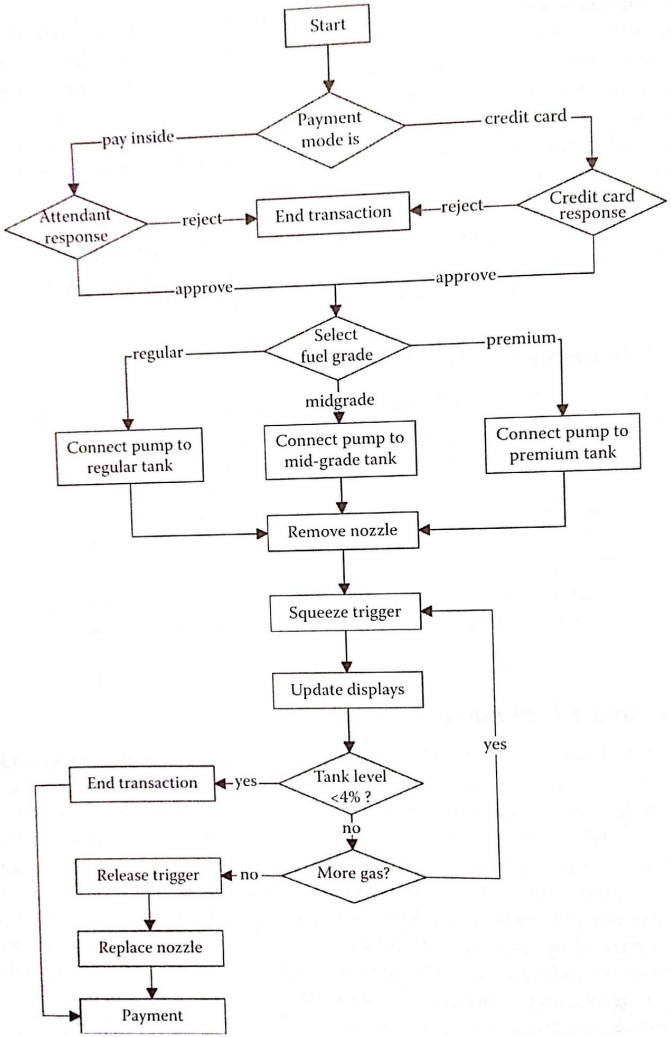
- Perhaps one of the earliest behavior models
- Self-explanatory, almost everyone knows about it



Smart Fuel Pump

Issues	Flowcharts
Sequence	yes
Selection	yes
Repetition	yes
Enable	no
Disable	no
Trigger	no
Activate	no
Suspend	no
Resume	no
Pause	no
Conflict	no
Priority	no

Issues	Flowcharts
Mutual Exclusion	no
Concurrent execution	no
Deadlock	no
Context-sensitive input events	no
Multiple-context output events	yes
Asynchronous events	no
Event quiescence	no



Advantages

- Ease of understanding
- Can be used to describe complex computations and algorithms
- High scalability.

Limitations

- Asynchronous and event-driven systems are hard to capture
- Almost no data representations.

Decision Tables

Overview

1. Table consists of rules, conditions, actions and results.

Stub	Rule 1	Rule 2	Rule 3
condition1	T	T	F
condition2	T	F	F
action1	x	x	...
action2	x	...	...
action3	...	...	x

Smart Fuel Pump

Issues	Flowcharts	Decision Tables
Sequence	yes	no
Selection	yes	yes
Repetition	yes	yes
Enable	no	no
Disable	no	no
Trigger	no	no
Activate	no	no
Suspend	no	no
Resume	no	no
Pause	no	no
Conflict	no	no
Priority	no	no

Table 5.22 Payment Mode Decision Table

c1. payment mode is	pay inside		credit card	
c2. attendant response is	approve	reject	–	–
c3. Credit card company response is	–	–	approve	reject
a1. reject transaction	–	X	–	X
a2. approve transaction	X	–	X	–
a3. perform select fuel grade table	X	–	X	–

Table 5.23 Fuel Grade Selection Decision Table

c1. fuel grade is	regular	mid-grade	premium
a1. connect to regular tank	X	–	–
a2. connect to mid-grade tank	–	X	–
a3. connect to premium tank	–	–	X
a4. remove nozzle	X	X	X
a5. perform fuel delivery table	X	X	X

Table 5.24 Fuel-Delivery Decision Table

c1. tank level is < 4%?	Yes	No	
c2. more gas?	–	Yes	No
a1. squeeze trigger	–	X	–
a2. release trigger	–	–	X
a3. update displays	X	X	X
a4. repeat table	–	X	X
a5. end transaction	X	–	X
a6. replace nozzle	X	–	X
a7. make payment	X	–	X

Smart Fuel Pump

Issues	Flowcharts	Decision Tables
Mutual Exclusion	no	yes
Concurrent execution	no	no
Deadlock	no	no
Context-sensitive input events	no	yes
Multiple-context output events	yes	yes
Asynchronous events	no	no
Event quiescence	no	no

Table 5.22 Payment Mode Decision Table

c1. payment mode is	pay inside		credit card	
c2. attendant response is	approve	reject	–	–
c3. Credit card company response is	–	–	approve	reject
a1. reject transaction	–	X	–	X
a2. approve transaction	X	–	X	–
a3. perform select fuel grade table	X	–	X	–

Table 5.23 Fuel Grade Selection Decision Table

c1. fuel grade is	regular	mid-grade	premium
a1. connect to regular tank	X	–	–
a2. connect to mid-grade tank	–	X	–
a3. connect to premium tank	–	–	X
a4. remove nozzle	X	X	X
a5. perform fuel delivery table	X	X	X

Table 5.24 Fuel-Delivery Decision Table

c1. tank level is < 4%?	Yes	No	
c2. more gas?	–	Yes	No
a1. squeeze trigger	–	X	–
a2. release trigger	–	–	X
a3. update displays	X	X	X
a4. repeat table	–	X	X
a5. end transaction	X	–	X
a6. replace nozzle	X	–	X
a7. make payment	X	–	X

Advantages

1. Works well for logic intensive applications.
2. Can be algebraically minimized

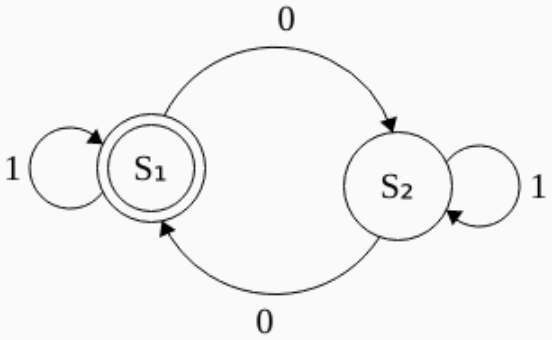
Limitations

1. Lack of order limits use for event driven applications.
2. Calculations can only be expressed as actions

Finite State Machines

Overview

- It is a directed graph
- Nodes are states
- Edges are transitions

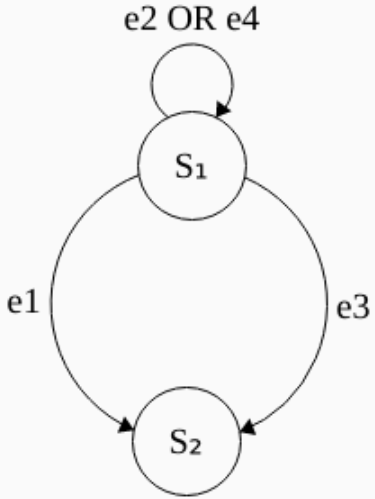


Overview contd.

- For transitions:
  - Numerators are events that cause transitions
  - Denominators are actions that are associated with the transition.

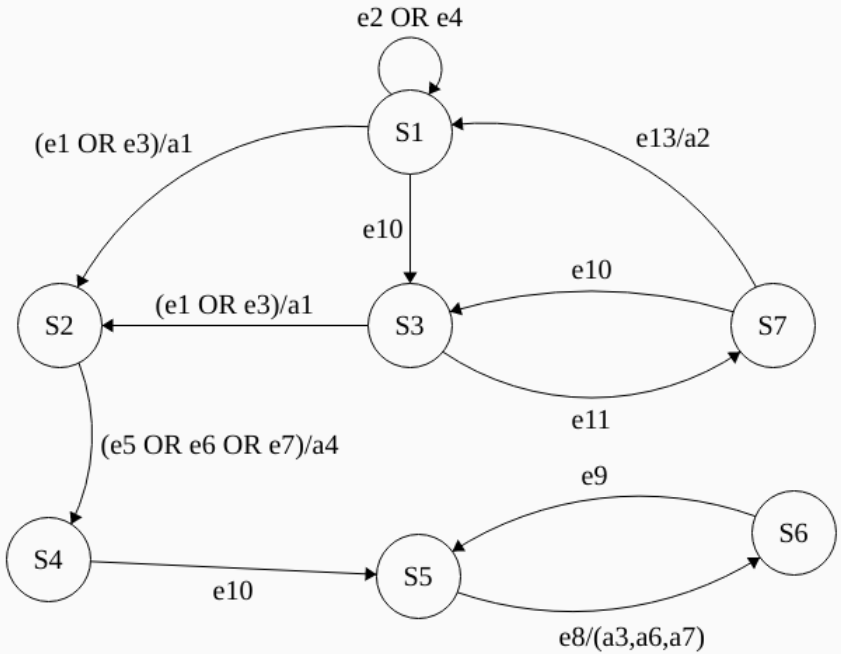
Smart Fuel Pump

- S1 - pump disabled
- S2 - pump enabled
- E1 - attendant approve
- E2 - attendant reject
- E3 - credit card approved
- E4 - credit card reject



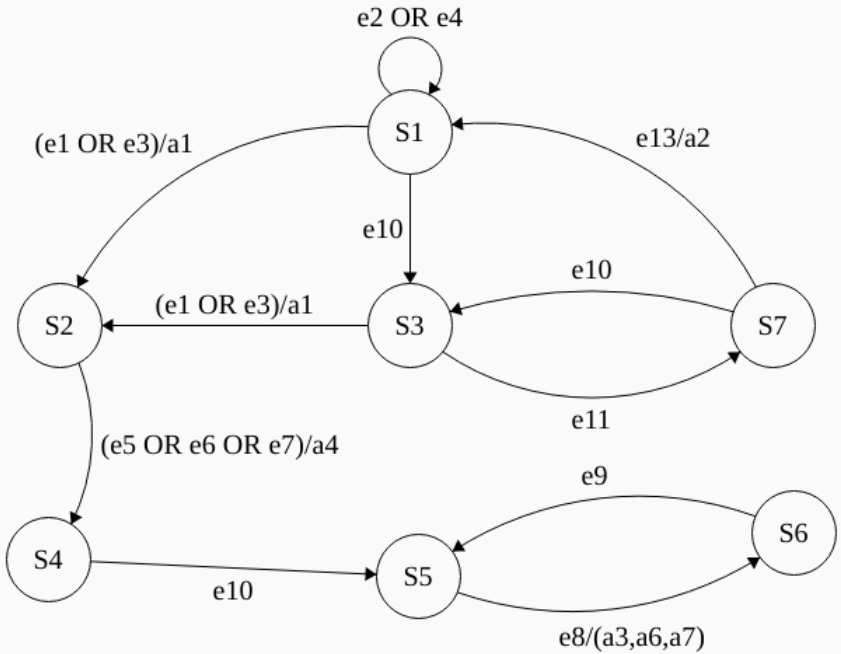
Smart Fuel Pump

- S1 - pump disabled
- S2 - pump enabled
- S3 - gun removed
- S4 - ready to pump
- S5 - trigger squeezed
- S6 - trigger released
- S7 - gun replaced



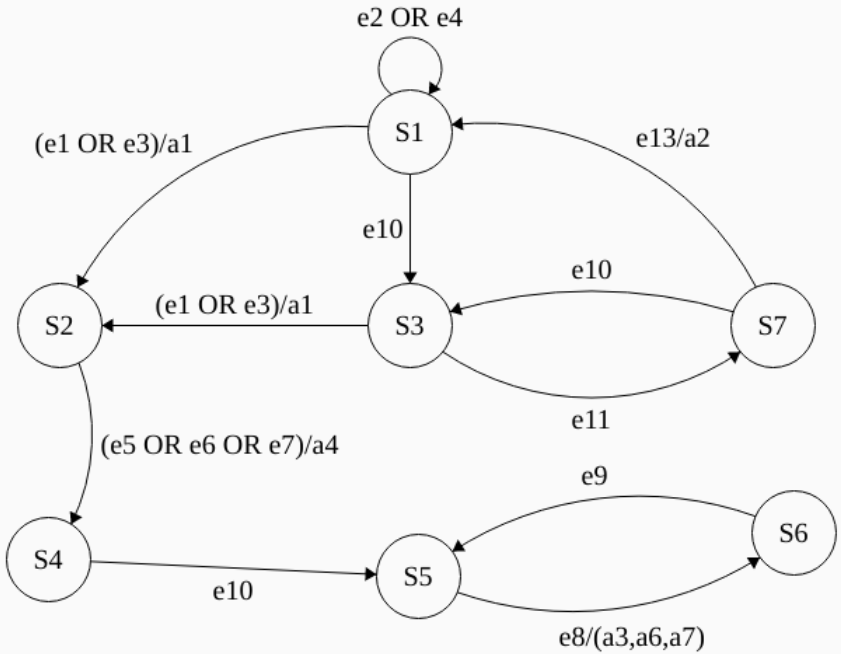
Smart Fuel Pump

- e1: attendant approve
- e2: attendant reject
- e3: credit card approve
- e4: credit card reject
- e5: select regular grade
- e6: select mid-grade
- e7: select premium grade
- e8: squeeze trigger
- e9: release trigger
- e10: remove gun
- e11: replace gun
- e12: tank level < 4%
- e13: customer ends fuel delivery



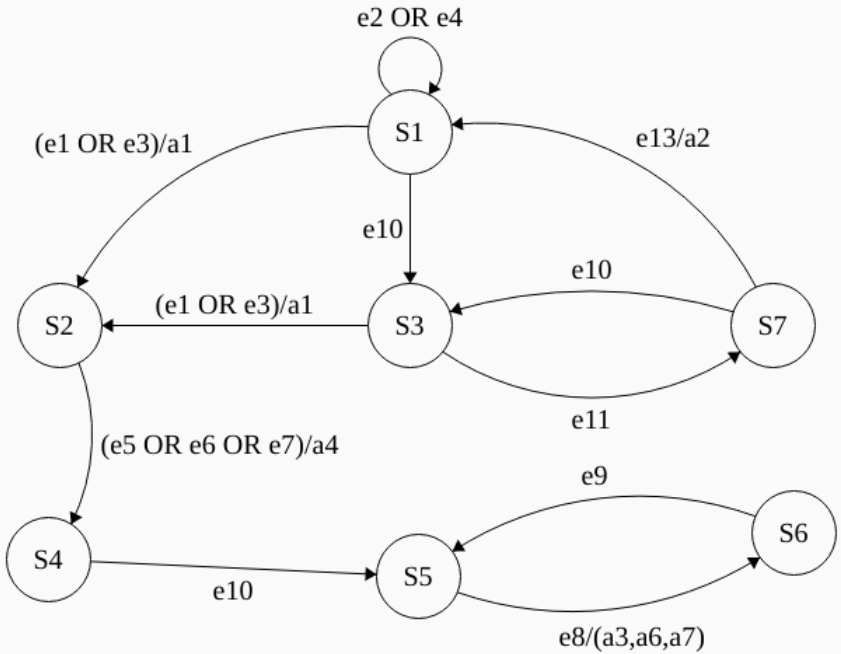
Smart Fuel Pump

- a1: start pump motor
- a2: stop pump motor
- a3: engage pump clutch
- a4: reset displays
- a5: free pump clutch
- a6: update volume pumped display
- a7: update transaction cost display



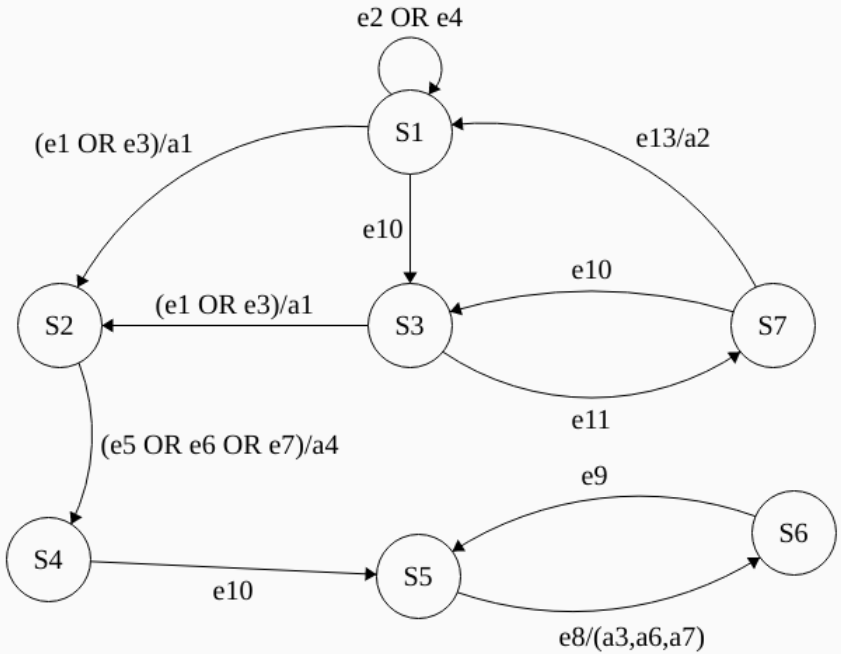
Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs
Sequence	yes	no	yes
Selection	yes	yes	no
Repetition	yes	yes	yes
Enable	no	no	no
Disable	no	no	no
Trigger	no	no	no
Activate	no	no	no
Suspend	no	no	no
Resume	no	no	no
Pause	no	no	no
Conflict	no	no	no
Priority	no	no	no



Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs
Mutual Exclusion	no	yes	no
Concurrent execution	no	no	no
Deadlock	no	no	no
Context-sensitive input events	no	yes	yes
Multiple-context output events	yes	yes	yes
Asynchronous events	no	no	no
Event quiescence	no	no	no



Advantages

- Intuitively understandable by developers and customers
- Paths can be easily seen, and parallelism is identifiable
- Attaching weights and probabilities to transitions can help in estimating costs for paths. Gives you more reasoning power.

Limitations

- Cannot represent a lot of behavioral issues. For e.g. selection.
- Finite machine explosion

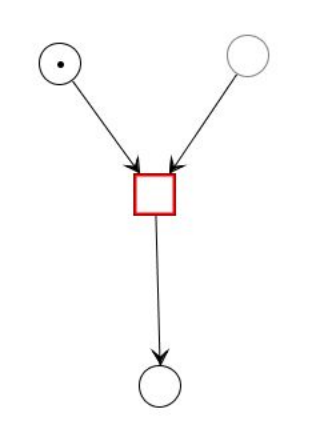
Petri Nets

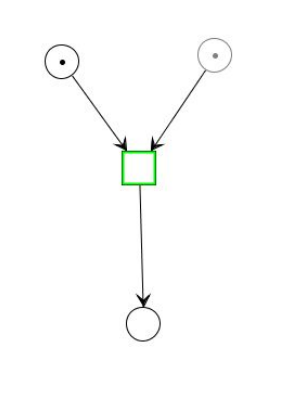
Overview

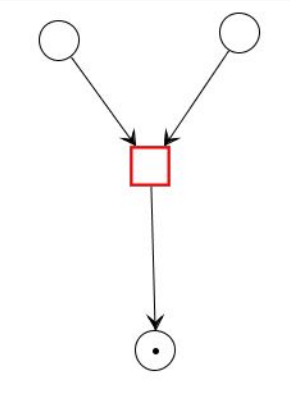
- Special form of directed graphs
- More commonly known as a bipartite graph
- There are two nodes, ***places*** and ***transitions***.

Overview

- A transition is enabled if there is at least one token in each of the input places.





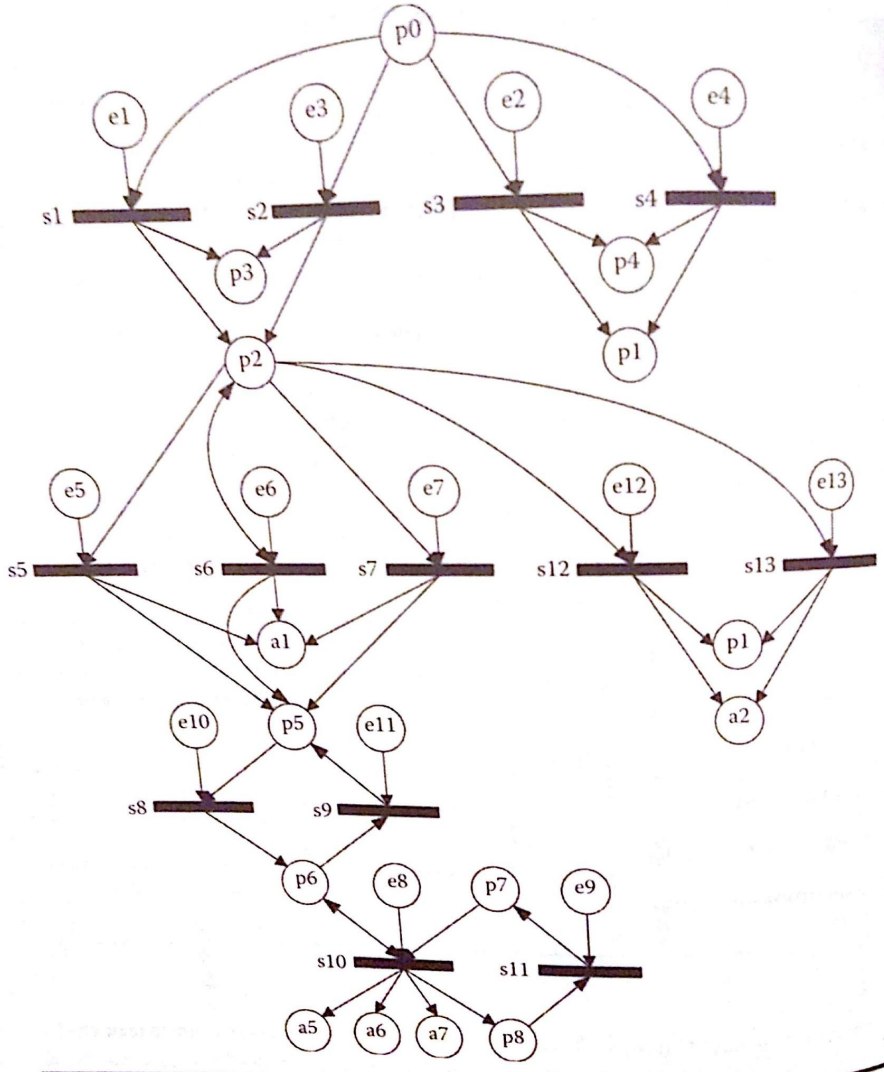


Examples

- In PNEditor

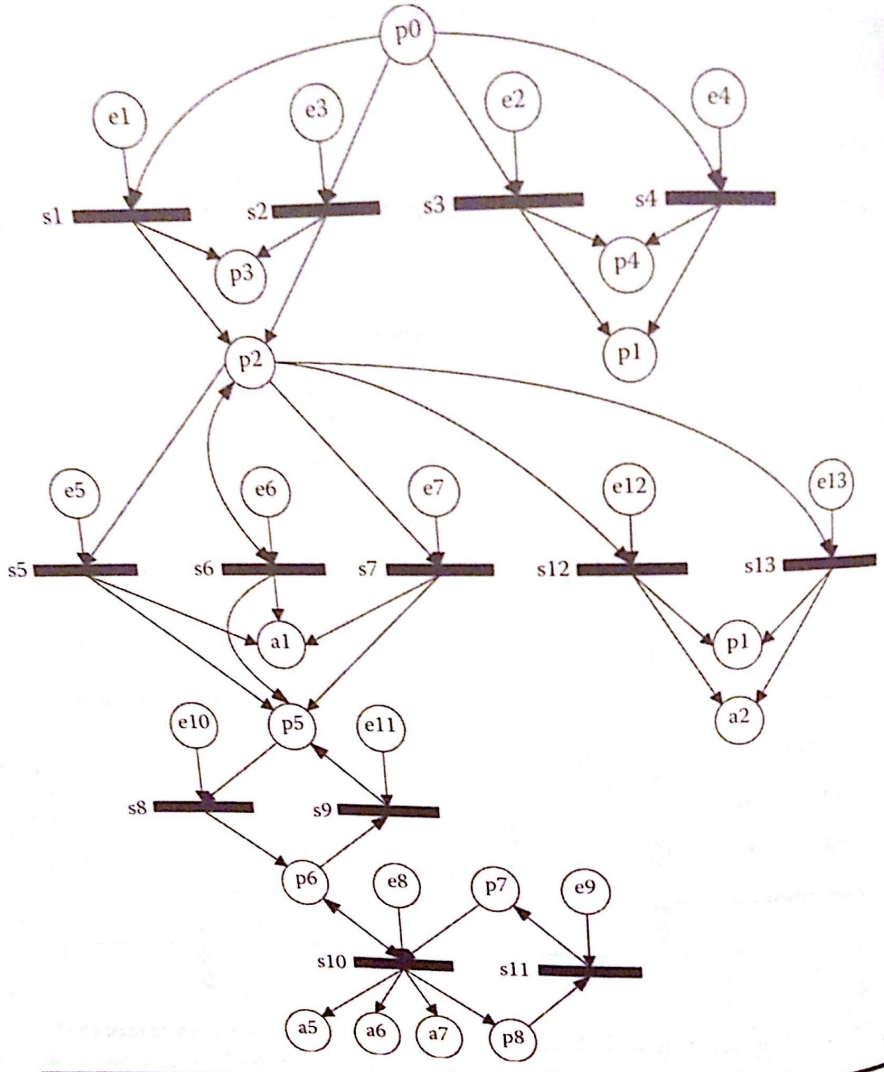
Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets
Sequence	yes	no	yes	yes
Selection	yes	yes	yes	yes
Repetition	yes	yes	yes	yes
Enable	no	no	no	yes
Disable	no	no	no	yes
Trigger	no	no	no	yes
Activate	no	no	no	yes
Suspend	no	no	no	yes
Resume	no	no	no	yes
Pause	no	no	no	yes
Conflict	no	no	no	yes
Priority	no	no	no	yes



Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets
Mutual Exclusion	no	yes	no	yes
Concurrent execution	no	no	no	yes
Deadlock	no	no	no	yes
Context-sensitive input events	no	yes	yes	Not nicely
Multiple-context output events	yes	yes	yes	Not nicely
Asynchronous events	no	no	no	Not nicely
Event quiescence	no	no	no	Not nicely



Advantages

- Extremely expressive
- Covers most behavioral issues

Limitations

- Graphical version does not scale up well for complex software
- Selection becomes more complex to model
- Does not model concurrency well

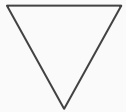
Event Driven Petri Nets

Overview

- Two slight enhancements over petri nets
- Differentiates between input and output events
- Consists of a different data place

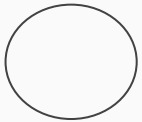
Overview

- Port input event

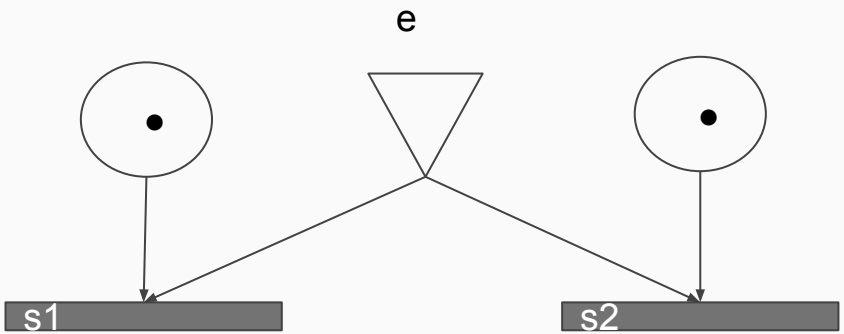


- Port output event

- Data place

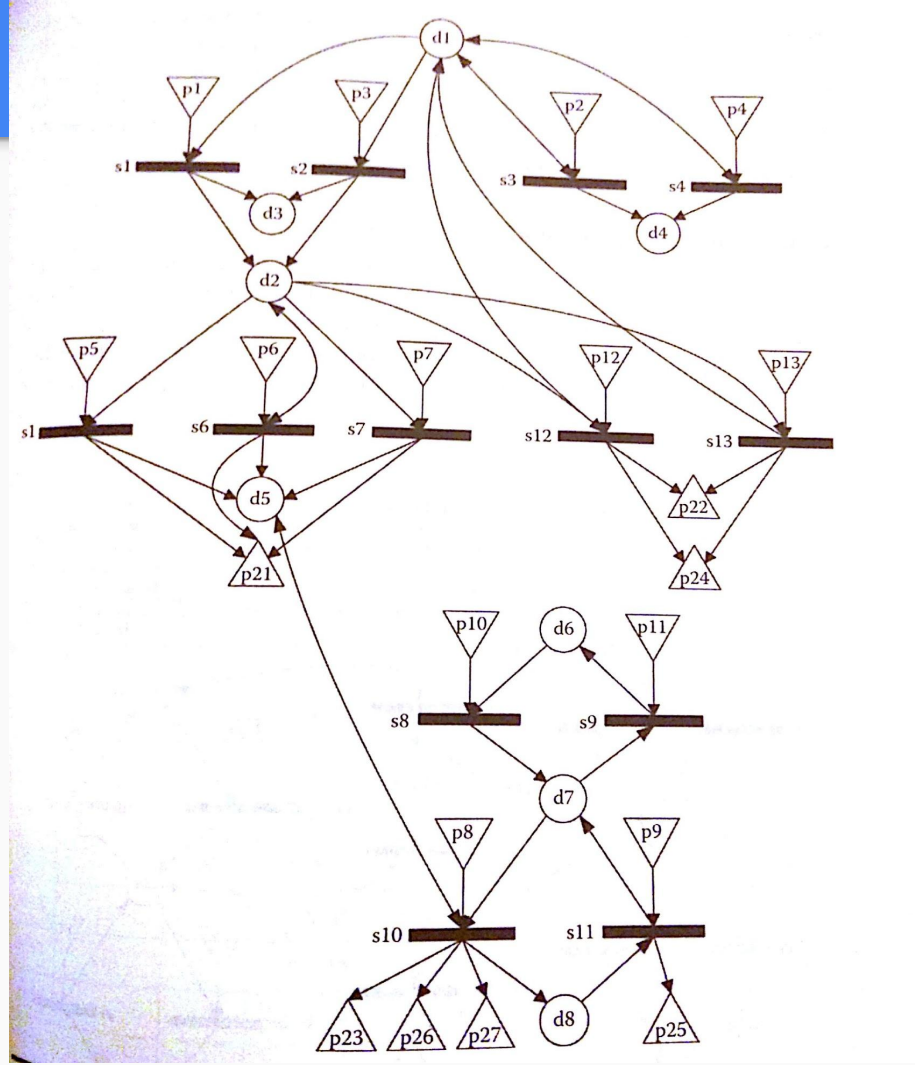


Example



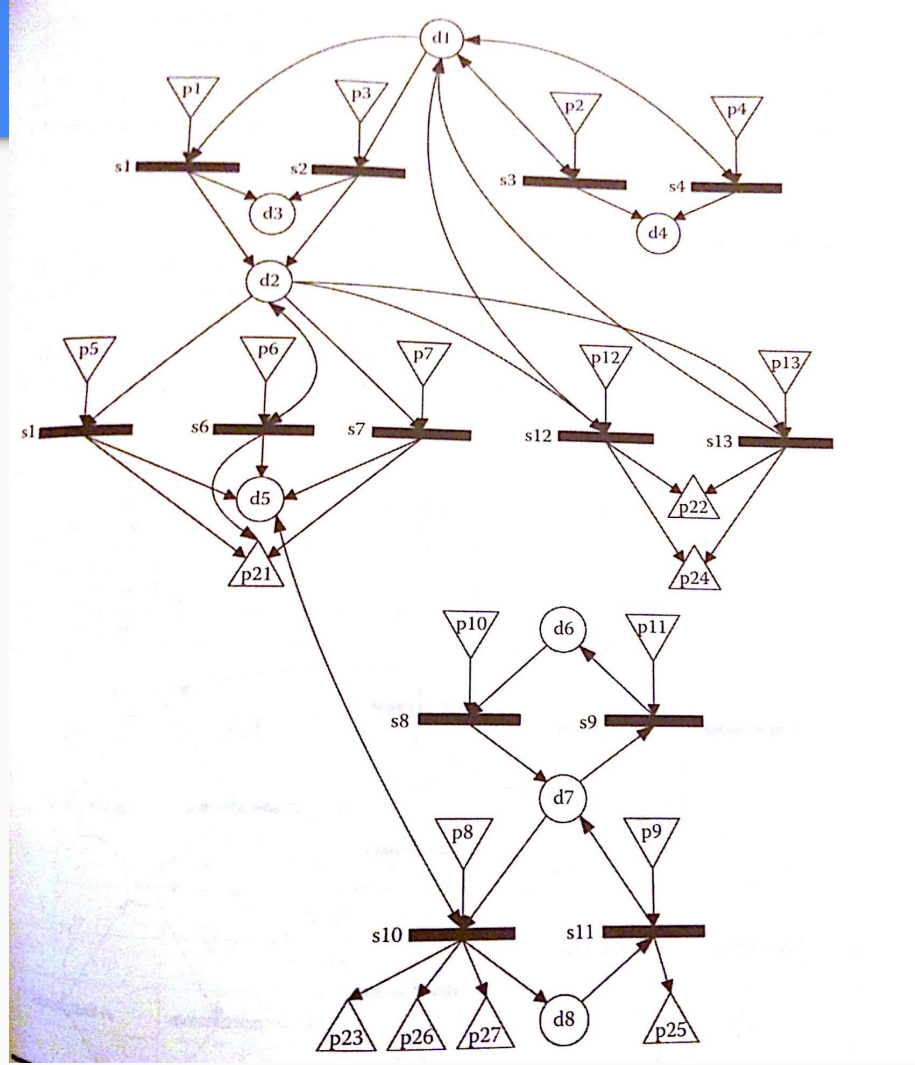
Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets	Event Driven Petri Nets
Sequence	yes	no	yes	yes	yes
Selection	yes	yes	yes	yes	yes
Repetition	yes	yes	yes	yes	yes
Enable	no	no	no	yes	yes
Disable	no	no	no	yes	yes
Trigger	no	no	no	yes	yes
Activate	no	no	no	yes	yes
Suspend	no	no	no	yes	yes
Resume	no	no	no	yes	yes
Pause	no	no	no	yes	yes
Conflict	no	no	no	yes	yes
Priority	no	no	no	yes	yes



Smart Fuel Pump

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets	Event Driven Petri Nets
Mutual Exclusion	no	yes	no	yes	yes
Concurrent execution	no	no	no	yes	yes
Deadlock	no	no	no	yes	yes
Context-sensitive input events	no	yes	yes	Not nicely	yes
Multiple-context output events	yes	yes	yes	Not nicely	yes
Asynchronous events	no	no	no	Not nicely	yes
Event quiescence	no	no	no	Not nicely	yes



Advantages

- All advantages of petri nets
- Addition of event places remove few original limitations
- Recognizes event-related issues such as context-sensitive input events

Limitations

- Same as petri nets except for event related problems

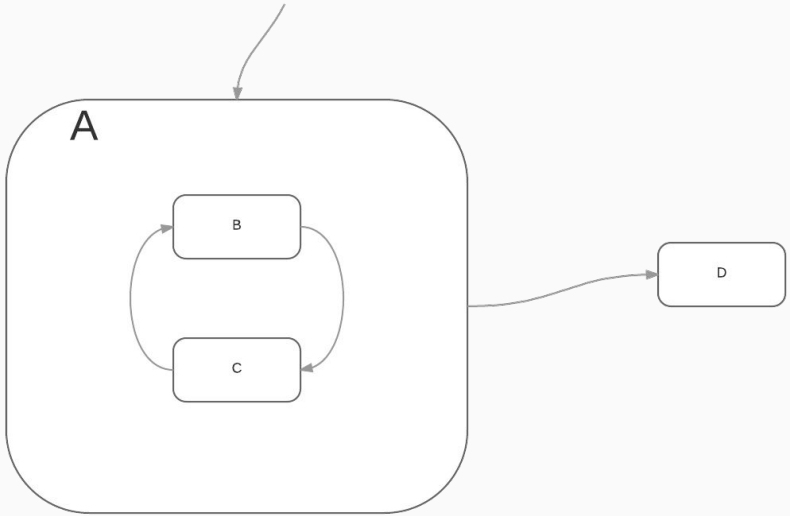
Statecharts

Overview

- A visual notation that combines:
  - The hierarchy of Venn diagrams
  - The connectedness of directed graphs
- An answer to the state explosions from Finite State Machines
- Commercially available tools. For e.g. StateMate (by IBM)

Notations

- Made up of blobs and edges
- Blobs are states
- Edges are transitions
- Starting states have an arrow towards them



Language Elements for Events

Event	Occurs when:
en(S)	State S is entered
ex(S)	State S is exited
st(A)	Activity A is started
sp(A)	Activity A is stopped
ch(V)	Value of V is changed
E1 and E2	E1 & E2 occur at the same time
E1 or E2	E1 or E2, or both, occur

Event	Occurs when:
tr(C)	Value of C is set to TRUE
fs(C)	Value of C is set to FALSE
rd(V)	Data V is read
wr(V)	Data V is written
tm(E,N)	N clock units passed since last time event E occured
E(C)	E has occured and condition C is TRUE
Not E	E didn’t occur

Language Elements for Conditions

Condition	TRUE when
in(S)	System is in state S
ac(A)	Activity A is active
hg(A)	Activity A is suspended
EXP1 R EXP2	Value of exp1 and exp2 satisfy relationship R
Not C	C is not TRUE
C1 and C2	C1 & C2 are TRUE
C1 or C2	C1 or C2, or both, are TRUE

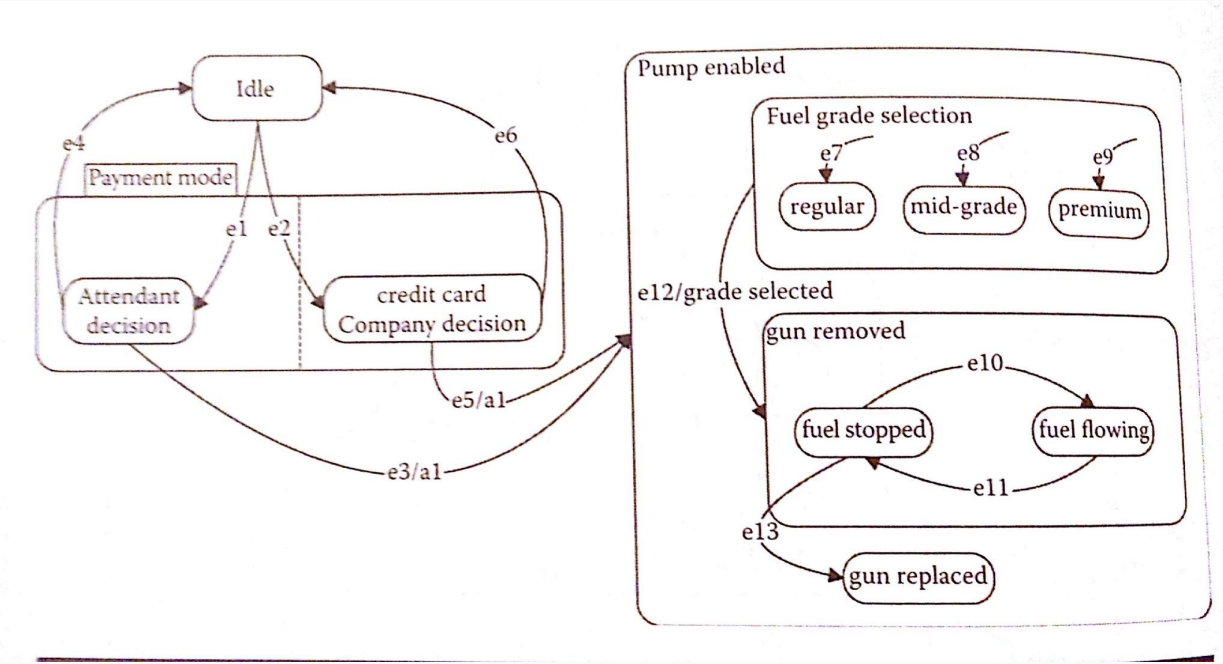
Language Elements for Actions

Action	Performs
E	Generate the event E
tr!(C)	Assign TRUE to cond. C
fs!(C)	Assign FALSE to cond. C
V:=EXP	Assign value EXP to V
st!(A)	Activate the activity A

Action	Performs
sp!(A)	Terminate the activity A
sd!(A)	Suspend the activity A
rs!(A)	Resume the activity A
rd!(V)	Read the value of V
wr!(V)	Write the value of V

Smart Fuel Pump

- e1: choose attendant pay
- e2: choose credit card pay
- e3: attendant approve
- e4: attendant reject
- e5: credit card approve
- e6: credit card reject
- e7: select regular grade
- e8: select mid-grade
- e9: select premium grade
- e10: squeeze trigger
- e11: release trigger
- e12: remove gun
- e13: replace gun
- e14: tank level < 4%
- e15: customer ends fuel delivery



Advantages

- Commercially available execution engines.
- High scalability for large, complex, concurrent applications.

Limitations

- Complexity of both notation and language on transitions.
- Makes it more difficult to perform technical inspection of a complex statechart.

Comparison

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets	Event Driven Petri Nets	Statechart
Sequence	yes	no	yes	yes	yes	yes
Selection	yes	yes	yes	yes	yes	yes
Repetition	yes	yes	yes	yes	yes	yes
Enable	no	no	no	yes	yes	yes
Disable	no	no	no	yes	yes	yes
Trigger	no	no	no	yes	yes	yes
Activate	no	no	no	yes	yes	yes
Suspend	no	no	no	yes	yes	yes
Resume	no	no	no	yes	yes	yes
Pause	no	no	no	yes	yes	yes
Conflict	no	no	no	yes	yes	yes
Priority	no	no	no	yes	yes	yes

Issues	Flowcharts	Decision Tables	FSMs	Petri Nets	Event Driven Petri Nets	Statechart
Mutual Exclusion	no	yes	no	yes	yes	yes
Concurrent execution	no	no	no	yes	yes	yes
Deadlock	no	no	no	yes	yes	yes
Context-sensitive input events	no	yes	yes	Not nicely	yes	yes
Multiple-context output events	yes	yes	yes	Not nicely	yes	yes
Asynchronous events	no	no	no	Not nicely	yes	yes
Event quiescence	no	no	no	Not nicely	yes	yes

Questions?

Activity

Smart Fuel Pump

- Draw a finite state machine for modeling the trigger squeeze/release event.
- http://madebyevan.com/fsm/
- Design a petri subnet for modeling trigger squeeze/release event.
- http://www.pneditor.org/
- Hint: Remember the resume/suspend primitive for petri net?

What are the differences?