Modeling and Simulation

Spatial Environments

Overview

- **Here we will understand about spatial environment in detail and it’s different types**

Spatial Environments

- Spatial environments in agent-based models generally have two variants:
  - Discrete spaces and continuous spaces.
- In a mathematical representation, in continuous spaces between any pair of points,
  - There exists another point,

Spatial Environments

- While in discrete spaces,
  - Though each point has a neighboring point, there do exist pairs of points without other points between them,
  - So that each point is separated from every other point.

ABM

- All continuous spaces must be implemented in ABM as approximations, so that continuous spaces are represented as discrete spaces
- Where the spaces between the points are very small.
- It should be noted that both discrete and continuous spaces can be either finite or infinite.

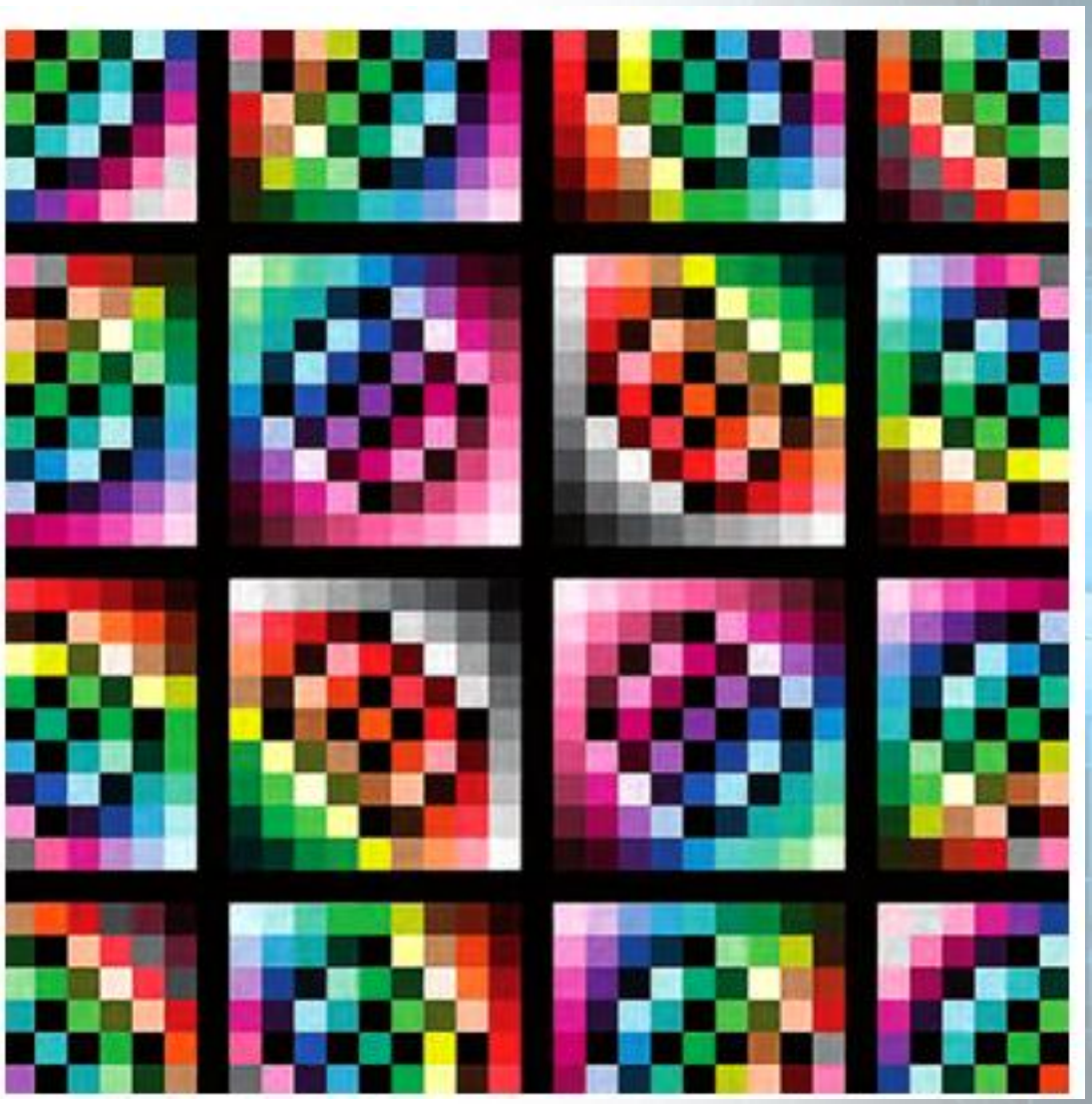
Discrete Spaces

- The most common discrete spaces used in ABM are lattice graphs (also sometimes referred to as mesh graphs or grid graphs),
- which are environments where every location in the environment is connected to other locations in a regular grid.

Toroidal square lattice

- For instance, every location in a toroidal square lattice has a neighboring location up, down, to the left, and to the right.
- As mentioned, the most common representation of the environment in *NetLogo is patches ,* which are located on a 2D lattice underlying the world of the ABM

Patches Displaying A Range Of Colors

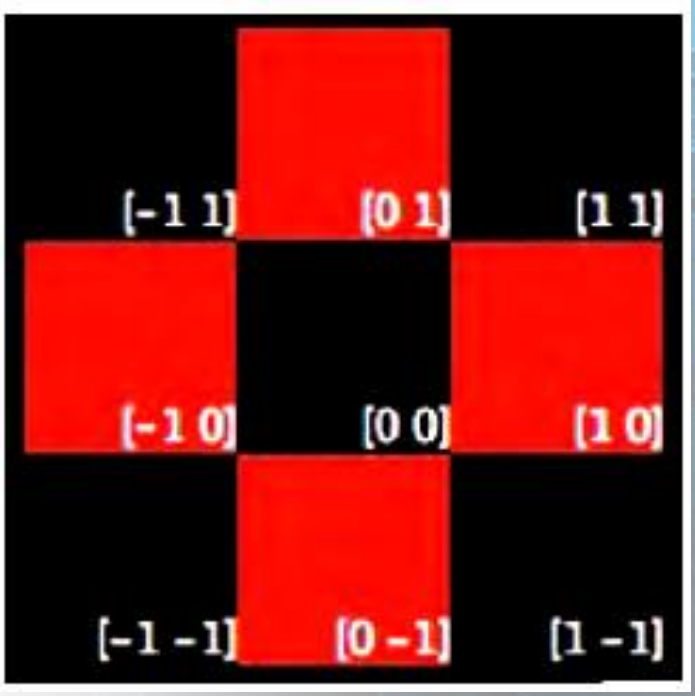


Square Lattices

- The square lattice is the most common type of ABM environment.
- A square lattice is one composed of many little squares,
- There are two classical types of *neighborhoods on a square lattice:*
  - Von Neumann neighborhood
  - Moore neighborhood

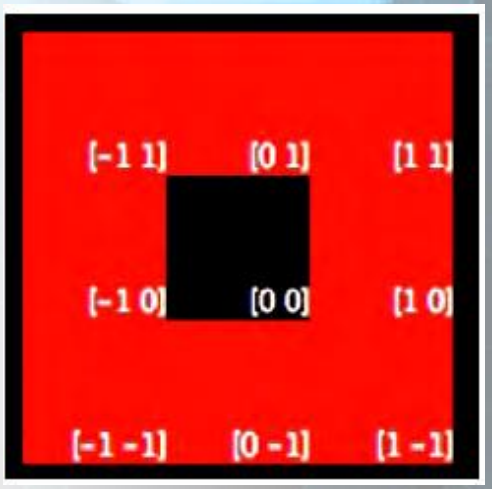
Von Neumann Neighborhood

- A von Neumann neighborhood of radius 1 is a lattice where each cell has four neighbors: up, down, left, and right



Moore Neighborhood

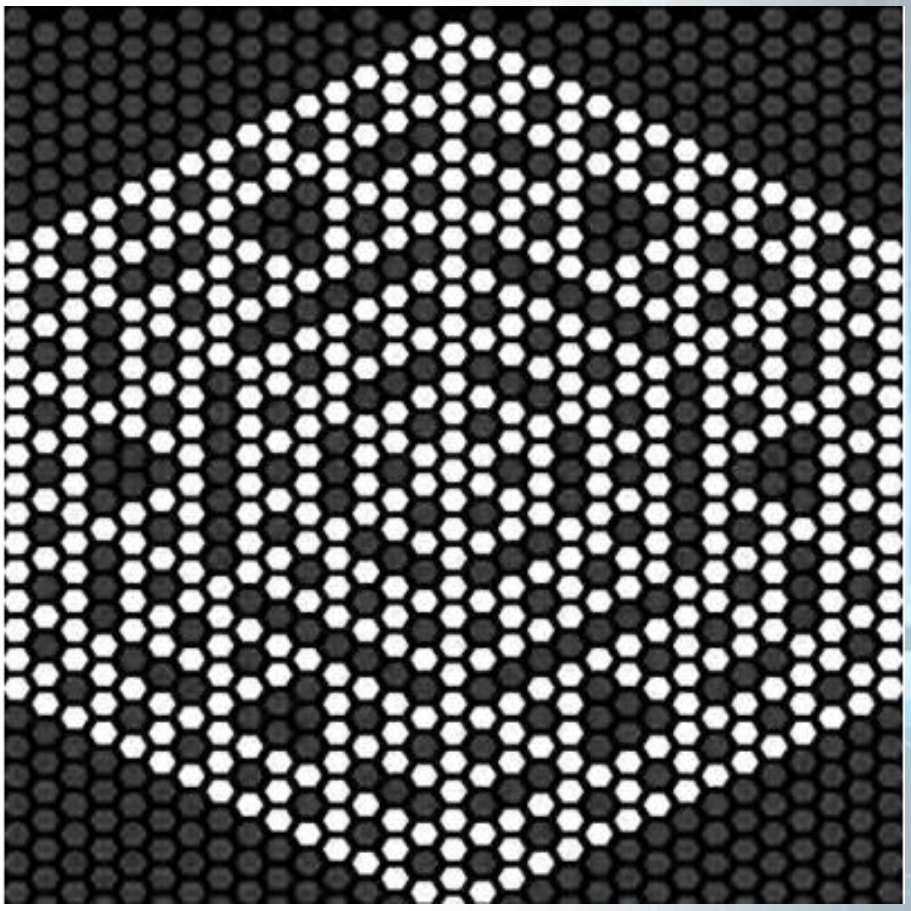
- **Moore neighborhood of radius 1 is a lattice in which each cell has eight neighbors in the eight directions that touch either a side or a corner: up, down, left, right, up-left, up-right, down-left, and down- right**



Hex Lattices

- A hex lattice has some advantages over square lattices.
- The center of a cell in a square grid is farther from the centers of some of the adjacent cells
- Many ABMs and ABM toolkits nevertheless employ square lattices.

Hex Cells example



In the Hex Cells example, each patch in the world maintains a set of six neighbors, with the agents located on each patch having a hexagonal shape.

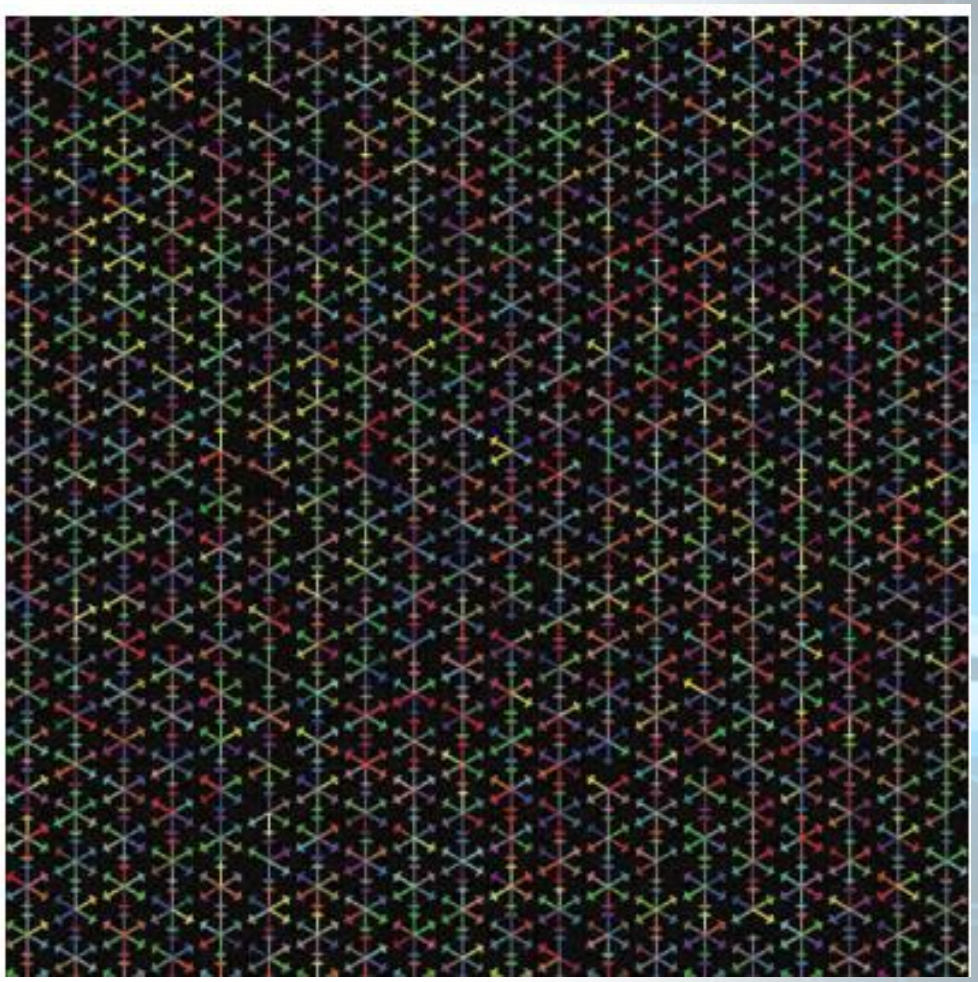
Continuous Spaces

- In a continuous space, there is no notion of a cell or a discrete region within the space.
- Instead, agents within the space are located at points in the space.
- These points can be as small as the resolution of the number representation allows.

Continuous Spaces

- In a continuous space, agents can move smoothly through the space, passing over any points in between their origin and destination,
- whereas in a discrete space, the agent moves directly from the center of one cell to another.
- In other words, all ABMs that use continuous spaces are actually using a very detailed discrete space

Hex Turtles example



NetLogo does not require that you specify whether you are using a continuous or discrete space ahead of time;

Boundary Conditions

- One other factor that comes into play when working with spatial environments is how to deal with boundaries, an issue for Hex and Square lattices as much as for continuous spaces.
- If an agent reaches a border on the far left side of the world and wants to go farther left, what happens?

Three Standard Approaches

- There are three standard approaches to this question, referred to as topologies of the environment:
  - It reappears on the far right side of the lattice (toroidal topology);
  - It cannot go any farther left (bounded topology)
  - It can keep going left forever (infinite plane topology).

Toroidal Topology

- A toroidal topology is one in which all of the edges are connected to another edge in a regular manner.
- Using a toroidal topology means that the modeler can ignore boundary conditions, which usually makes model development easier.

Bounded Topology

- A bounded topology is one in which agents are not allowed to move beyond the edges of the world.
- This topology is a more realistic representation of some environments.
- For instance, in the Traffic Basic model, where the cars only drive from left to right, the top and bottom of the world are bounded

Infinite-plane Topology

- An infinite-plane topology is one where there are no bounds. In other words , agents can keep moving and moving in any direction forever
- Some ABM toolkits provide built-in support for infinite plane topologies, NetLogo does not

Summary

- In most cases, however, a toroidal or bounded topology will be the more appropriate (and simpler) choice to implement.
- Credits : Uri Wilensky

Modeling and Simulation

Network-Based Environments

Overview

- **Here we will understand about Network-Based Environments**

Link

- A link is defined by the two ends it connects, which are frequently referred to as nodes
- We will use the network/node/link vocabulary throughout Module
- In NetLogo, links are their own agent-type.

lattice networks

- lattice graphs can also be called lattice networks , with the property that each position in the network looks exactly like every other position in the network
- ABM environments usually do not implement lattice environments as networks for both conceptual and efficiency reasons

Random Networks

- Besides the regular networks described earlier, the three most common network topologies are random , scale-free , and small-world
- In random networks , each individual is randomly connected to other individuals.

Random Networks

- For example, if you had a model of agents moving around a large room and connected every agent in the room to another agent based on which agent had the next largest last two digits of their social security number, you would probably create a random network.

A Random Network

- Here we show one simple methods for creating a random network.
- This code is also in the Random Network model in the chapter 5 subfolder of the IABM Text-book folder of the NetLogo models library:

Code for creating a random network

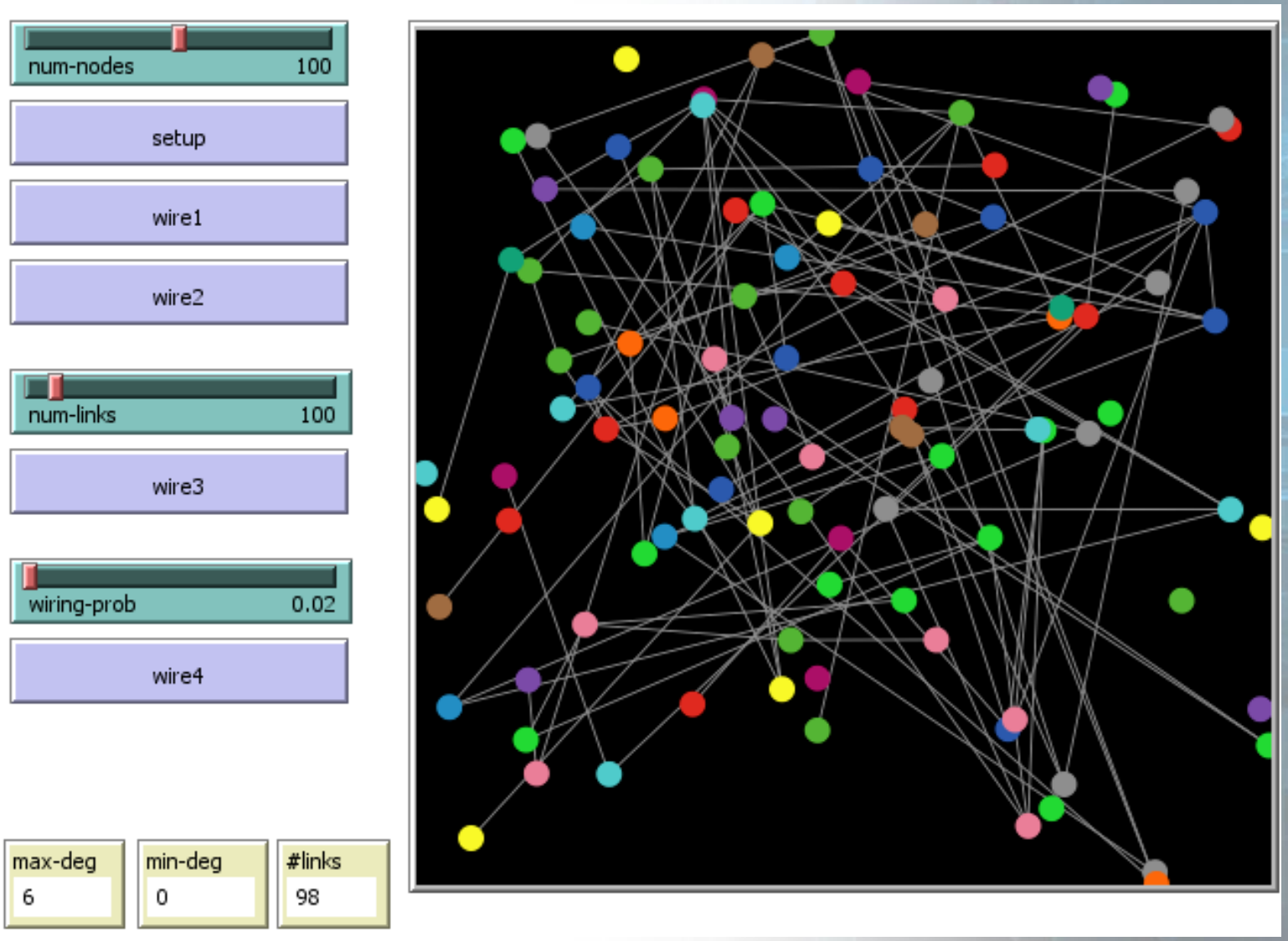
```
;; Create nodes.
to setup
  clear-all
  create-turtles num-nodes [
    set shape "circle"
    setxy random-xcor random-ycor
  ]
  reset-ticks
end

;; Ask each node to create a link with a random other node.
;; Results in "just under" NUM-NODES links because no new link is created
;; if a node tries to connect to a node with which it is already linked.
to wire1
  ask links [ die ]
  ask turtles [
    create-link-with one-of other turtles
  ]
end

;; Pick a random node and ask it to create a link with a random other node.
;; Results in "just under" NUM-NODES links because no new link is created
;; if a node tries to connect to a node with which it is already linked.
to wire2
  ask links [ die ]
  repeat num-nodes [
    ask one-of turtles [
      create-link-with one-of other turtles
    ]
  ]
end

;; This is the classic Erdős-Rényi random network.
;; It uses WHILE to ensure we get NUM-LINKS links.
to wire3
  ask links [ die ]
  if num-links > max-links [ set num-links max-links ]
  while [ count links < num-links ] [
```

Random Network model



Scale-free Networks

- **Scale-free networks have the property that any sub network of the global network has the same properties as the global network**
- **A common way to create this type of *network is by adding new nodes and links to the system so that extant nodes with a large number of links are more likely to receive new connections***

Preferential Attachment

- This technique is sometimes called preferential attachment , since nodes with more connections are attached to preferentially.
- This method for network creation tends to produce networks with central nodes that have many radiating links;

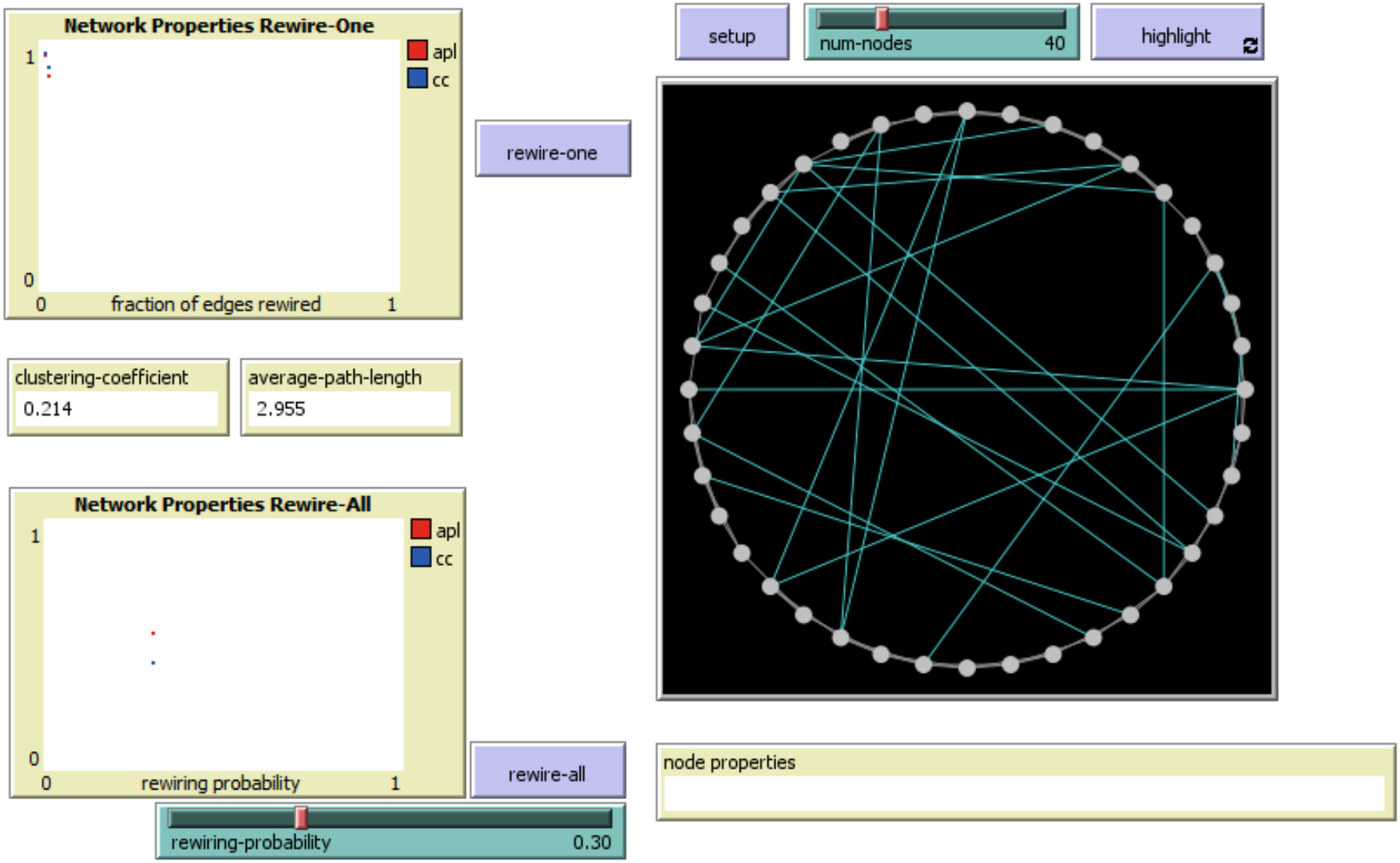
Small-world Network

- The final standard network topology is known as a small-world network.
- Small-world networks are made up of dense clusters of highly interconnected nodes joined by a few long distance links between them.

Small-world model

- **Small-world networks are sometimes created by starting with regular networks**
- **There are many ways to characterize networks.**
- **Two commonly used ways are average path length and clustering coefficient**
- ***The average path length is the average of all the pair wise distances between nodes in a network*.**

Small-world model



Small-World model in the Networks section of the NetLogo models library (Wilensky, 2005a)

Average Path Length

- The average path length is the average of all the pair wise distances between nodes in a network.
- In other words, we measure the distance between every pair of nodes in the network and then average the results.
- Average path length characterizes how far the nodes are from each other in a network.

Clustering Coefficient of a Network

- The clustering coefficient of a network is the average fraction of a node's immediate neighbors who are also neighbors of the node's other neighbors
- In networks with a high average clustering coefficient, any two neighboring nodes tend to share many of their neighbors in common,

Summary

- NetLogo also includes a special extension, the network extension, for creating, analyzing, and working with networks.
- This extension enables full integration of network theory methods into ABM.
- Credits : Uri Wilensky

Modeling and Simulation

Special Environments

Overview

- **Here we will understand about two of the most interesting topologies involve the use of 3D worlds and Geographic Information Systems (GIS).**

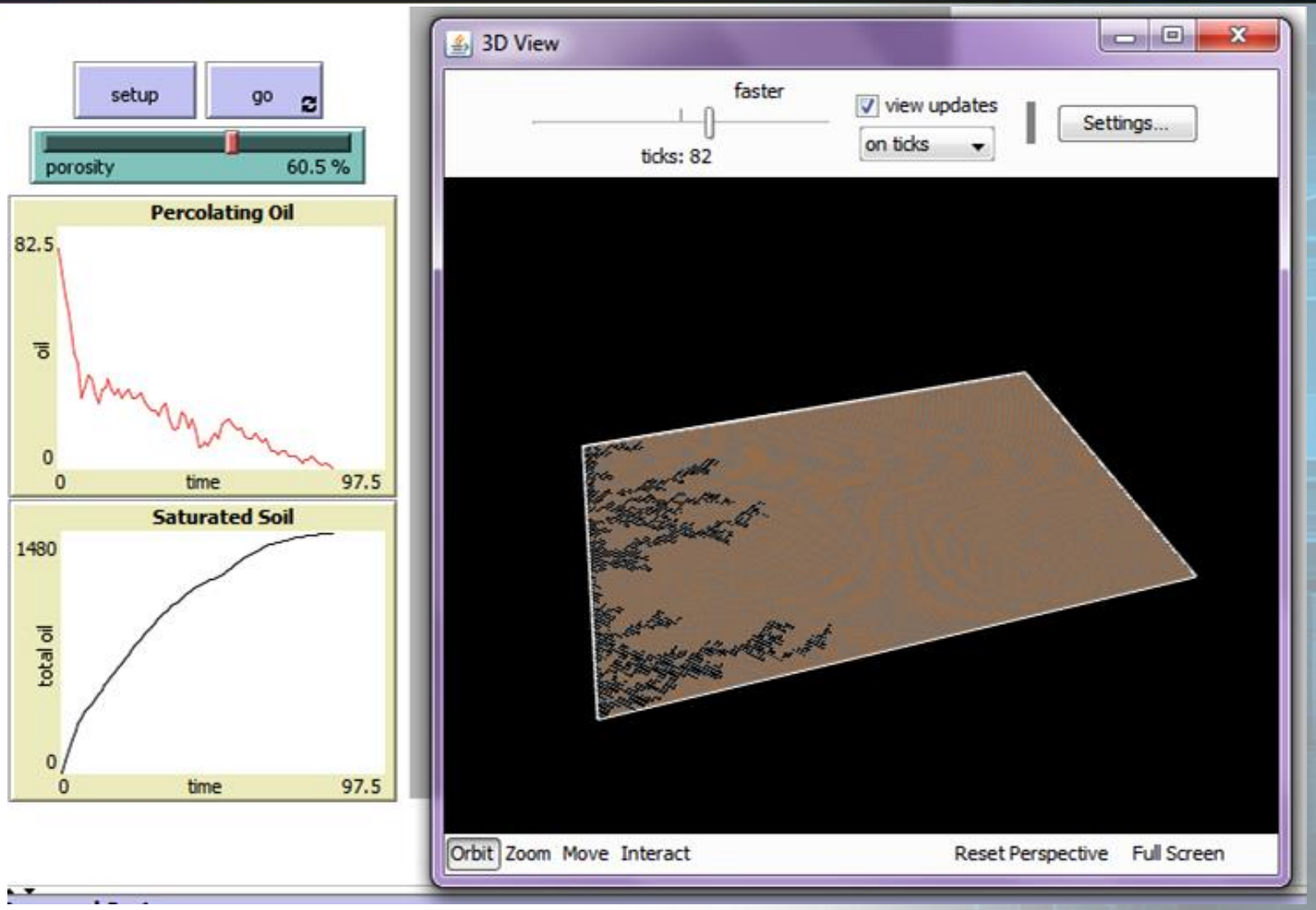
3D Worlds

- 3D environments enable model developers to explore complex systems which are irreducibly bound up with a third dimension as well as sometimes increasing the apparent physical realism to their models.
- There is a version of NetLogo called NetLogo 3D

Working With 3D Worlds

- Working with 3D worlds is not much different from working with 2D worlds.
- Although, using a 3D world does require additional data and commands.
- For instance, agents now have a Z coordinate, and new commands are needed to manipulate this new degree of freedom.

Percolation 3D



There is a three-dimensional version of the Percolation model

Code for PERCOLATE procedure in 2D

```
to percolate
    ask current-row with [pcolor = red] [
       ;; oil percolates to the two patches southwest and southeast
       ask patches at-points [[-1 -1] [1 -1]]
           [ if (pcolor = brown) and (random-float 100 < porosity)
              [ set pcolor red ] ]
       set pcolor black
       set total-oil total-oil + 1
    ]
    ;; advance to the next row
    set current-row patch-set [patch-at 0 -1] of current-row
end
```

The code for the PERCOLATE procedure in 2D Percolation looks like this:

Code for PERCOLATE procedure in 3D

```
to percolate
    ask current-row with [pcolor = red] [
       ;; oil percolates to the four patches one row down in the z-coordinate and
    ;; southwest, southeast, northeast, and northwest
       ask patches at-points [[-1 -1 -1] [1 -1 -1] [1 1 -1] [-1 1 -1]]
           [ if (pcolor = black) and (random-float 100 < porosity)
              [ set pcolor red ] ]
       set pcolor brown
       set total-oil total-oil + 1
    ]
    ;; advance to the next row
    set current-row patch-set [patch-at 0 0 -1] current-row
end
```

Notice that in the 2D version, the code has each patch look at patches that are below, left, and right of the patch. This will change in the 3D version. In 3D Percolation (Wilensky& Rand, 2006), PERCOLATE looks like this:

The difference between 3D percolate and 2D

- The only difference between the 3D percolate and its 2D counterpart is that instead of percolating a line of patches at a time, We percolate a square of patches.
- As a result, each patch must ask four patches below it (in a cross shape) to percolate, not just two as it does in the 2D version.

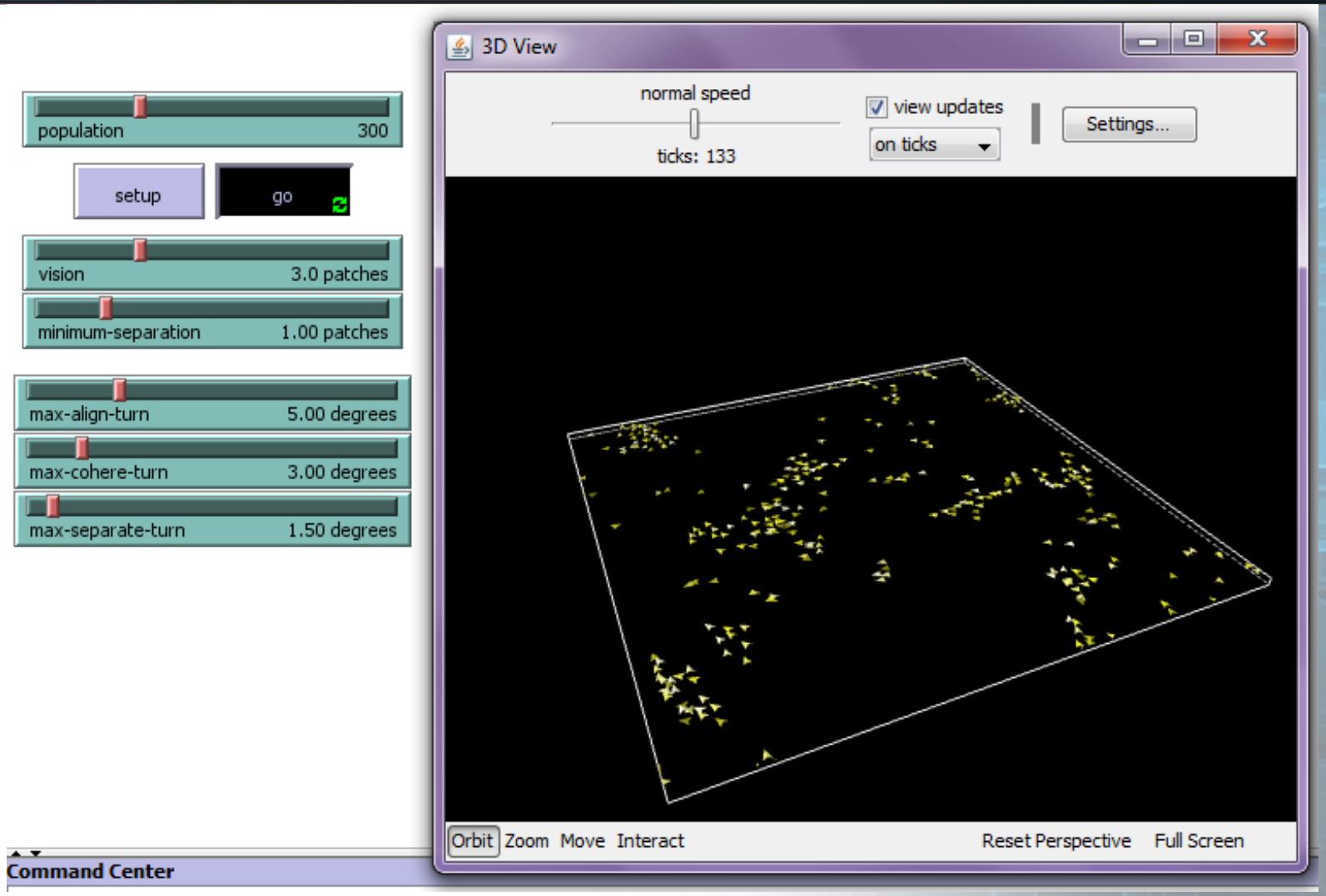
GIS-Based

- Geographic Information Systems (GIS) are environments that record large amounts of data that are related to physical locations in the world.
- GISs are widely used by environmental scientists to organize data and make decisions about any large area of land.

GIS Tools And Techniques

- GIS tools and techniques allow for a more in-depth exploration of the pattern of a complex system.
- Agents moving on a GIS terrain may be constrained to interact on that terrain.
- Thus, GIS can serve as an interaction topology

The 3D model, Flocking 3D



Where does ABM fit in to this picture? GIS can provide an environment for an ABM to operate in.

Summary

- Using the NetLogo GIS extension is the preferred method of importing large amounts of GIS data into a NetLogo model
- Credits: Uri Wilensky

Modeling and Simulation

Interactions

Overview

- **We will look at how agents and environments interact.**
  - **There are five basic classes of interactions that exist in ABMs:**
  - ***Agent-self ,***
  - ***Environment-self ,***
  - ***Agent-agent ,***
  - ***Environment - environment ,***
  - ***And finally, agent-environment .***

Agent-Self Interactions

- Agents do not always need to interact with other agents or the environment.
- In fact, a lot of agent interaction is done within the agent.
- For instance, most of the examples of advanced cognition that we discussed in the Agent section involve the agent interacting with itself.

Birth Events are a Typical Event In ABMs

```
;; check to see if this agent has enough energy to reproduce
to reproduce
    if energy > 200 [
       set energy energy - 100  ;; reproduction costs energy to the parent
       hatch 1 [ set energy 100 ] ;; which is transferred to the offspring
    ]
end
```

In above code, the agent considers its own state and, based on this state, decides whether or not to give birth to a new agent.

It then manipulates its state, lowering its energy and creating the new agent. This is the typical way of having agents reproduce:.

Environment-Self Interactions

- Environment-self interactions are when areas of the environment alter or change themselves.
- For instance, they could change their internal state variables as a result of calculations.

Example of an Environment-self Interaction

```
;; regrow the grass
to regrow-grass
    ask patches [
       set grass-amount grass-amount + grass-regrowth-rate
       if grass > 10 [
           set grass 10
                          ]
recolor-grass
    ]
end
```

The classic example of an environment-self interaction is when the grass re-grows:

Each patch is asked to examine its own state and increment the amount of grass it has, but if it has too much grass then it is set back to the maximum value it can contain.

Agent-Agent Interactions

- Interactions between two or more agents are usually the most important type of action within agent-based models.
- We saw a canonical example of agent-agent interactions in the Wolf Sheep Predation model when the wolves consume the sheep:

Example Of Agent-agent Interactions

```
;; wolves eat sheep
to eat-sheep
    if any? sheep-here [ ;; if there are sheep here then eat one
       let target one-of sheep-here
       ask target [
           die
       ]
       ;; increase the energy by the parameter setting
       set energy energy + energy-gain-from-sheep
    ]
end
```

One agent is consuming another agent and taking its resources, whereby the wolf always eats the sheep.

However, it is also possible to add competition or flight to this model, where the wolf gets a chance of eating the sheep and the sheep gets a chance to flee. *Competition is another example of agent-agent interaction.*

Communication

- **A final example of *agent-agent interaction is communication*. Agents can share information about their own state as well as that of the world around them.**
- **This type of interaction allows agents to gain information to which they might not have direct access.**

Environment-Environment Interactions

- **Interactions between different parts of the environment are probably the least commonly used type of interaction in agent-based models.**
  - ***However, there are some common uses of environment-environment interactions: one of these is diffusion.***

Ants model

```
diffuse chemical (diffusion-rate / 100)
ask patches
    [ set chemical chemical * (100 - evaporation-rate) / 100
;; slowly evaporate chemical
       recolor-patch ]
```

In the Ants model, the ants place a pheromone in the environment, which is then diffused throughout the world via an environment-environment interaction. This interaction is contained in the above piece of code in the main GO procedure:

Agent-Environment Interactions

- Agent-environment interactions occur when the agent manipulates or examines part of the world in which it exists, or when the environment in some way alters or observes the agent.
- A common type of agent-environment interaction involves agents observing the environment.

Ants model

```
to look-for-food ;; turtle procedure
    if food > 0
    [ set color orange + 1      ;; pick up food
        set food food - 1                ;; and reduce the food source
        rt 180                                    ;; and turn around
    stop ]
    ;; face in the direction where the chemical smell is strongest
    if (chemical >= 0.05) and (chemical < 2)
    [ uphill-chemical ]
end
```

The Ants model demonstrates this kind of interaction when the ants examine the environment to look for food and sense pheromone:

If there is food, the ant then picks up the food and turns around back to the nest, and the procedure stops.

Agent Movement

- Another common type of agent-environment interaction is agent movement. In some ways, movement is simply an agent-self interaction, since it only alters the current agent's state.
- In the Ants model the ants move around by " wiggling "

Summary

- We have reviewed here the five basic different types of interactions
- we have covered here some examples of their most common applications.
- Credits : Uri Wilensky

Modeling and Simulation

Observer/User Interface

Overview

- Now that we have talked about the agents, the environments, and the interactions that occur between agents and environmental attributes, here we will discuss who controls the running of the model

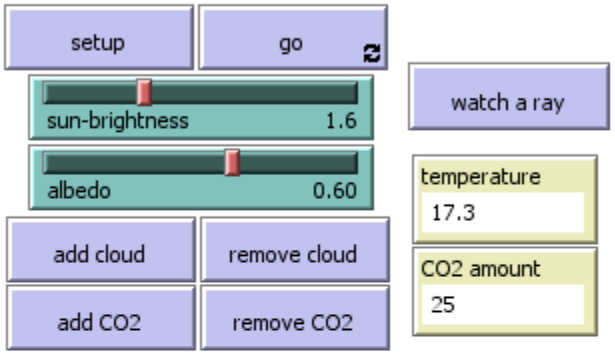
Observer

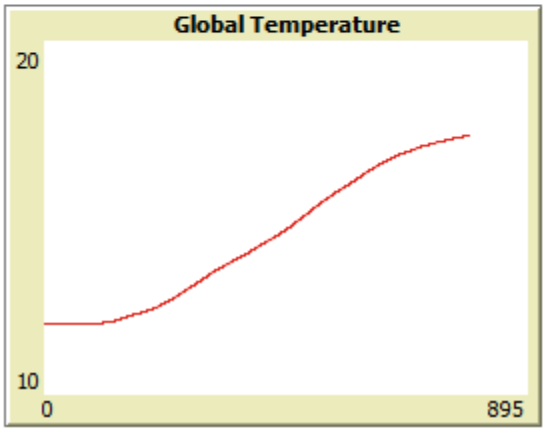
- The observer is a high-level agent that is responsible for
  - Ensuring that the model runs and proceeds according to the steps developed by the model author.
  - The observer issues commands to agents and the environment,

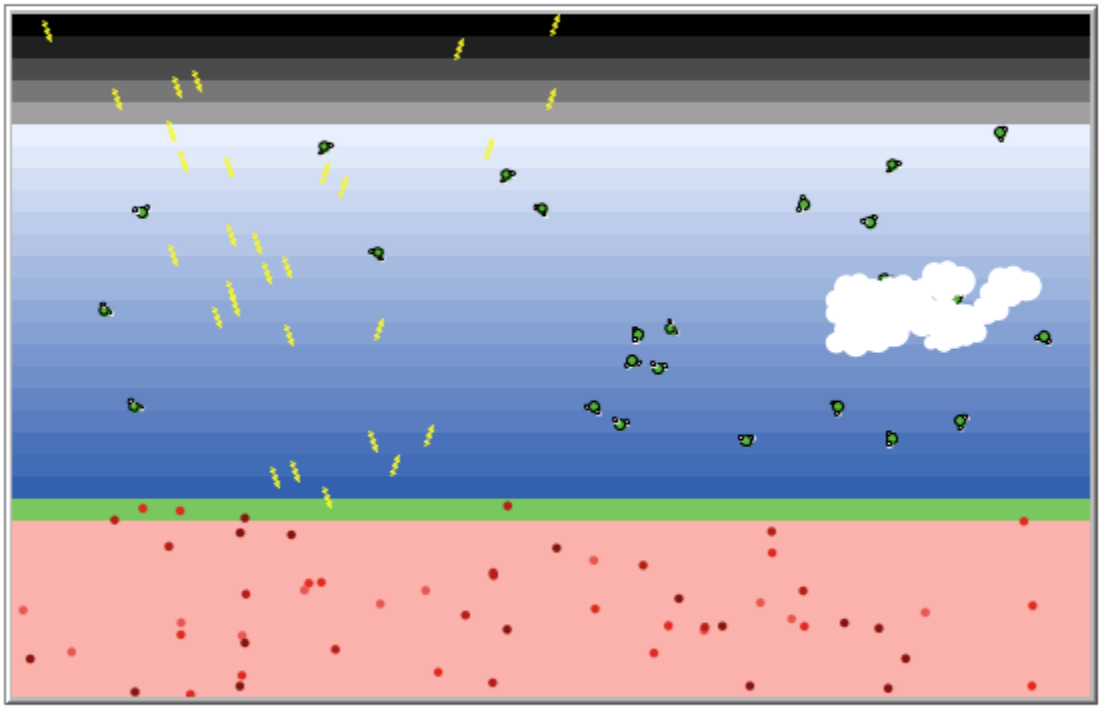
Properties Specific to the observer

- The only properties that one could consider to be specific to the observer are those relating to the perspective from which the modeled world is viewed
- In NetLogo the view may be centered on a specific agent

Climate Change model







In NetLogo the view may be centered on a specific agent, or focusing a highlight on a certain agent, using the FOLLOW, WATCH, or RIDE commands

User Input and Model Output

- ABMs require a control interface or a parameter group that allows the user to setup different parameters and settings for the ABM
- The most common control *mechanism is a button* , which executes one or more commands within the model
  - *if it is a forever button it will continue*

Command Center

- The command center is a very useful feature of NetLogo, as it allows the user to interactively test out commands, and manipulate agents and the environment.
- The *input controls include* sliders, switches, choosers, and input boxes.
- The *output controls consist* of monitors, plots, an output area, and notes.

Sliders, Switches & Input Boxes

- Sliders enable the model user to select a particular value from a range of numerical values
- Switches enable the user to turn various elements of a model off or on
- Input boxes are more free-form, allowing the user to input text that the model can use

Monitors , Plots & Notes

- Monitors display the value of a global variable or calculation updated several times a second.
- Plots provide traditional 2D graphs enabling the user to observe the change of an output variable over time
- Notes enable the modeler to place text information on the Interface tab

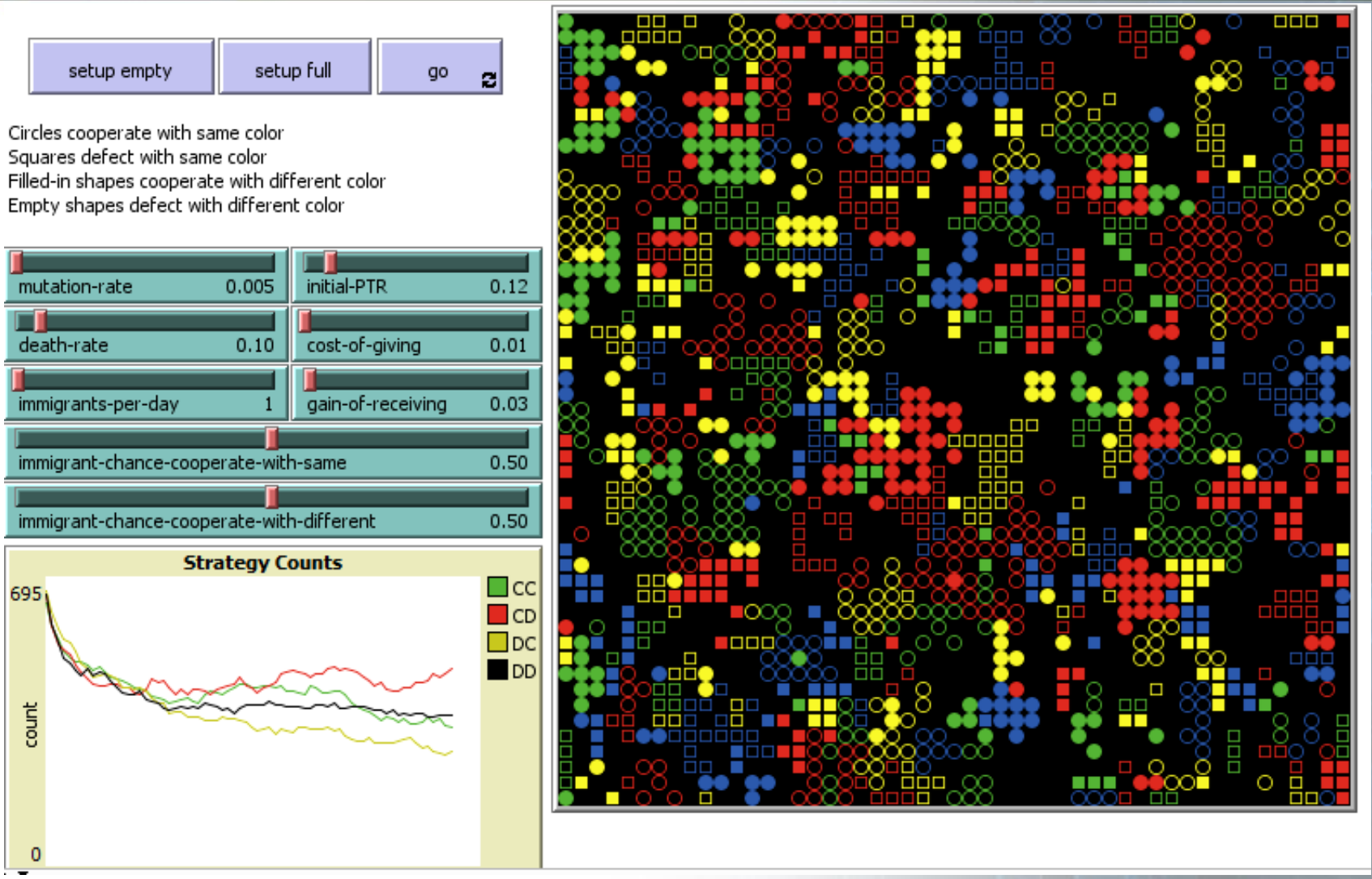
Visualization

- Visualization is the part of model design concerned with how to present the data contained in the model in a visual way
- There are three guidelines that should be kept in mind whenever designing the visualization of an ABM:
  - Simplify, Explain, and Emphasize

Guidelines for Designing Visualization

- **Simplify the visualization:**
  - ***Make the visualization as simple as possible***
- **Explain the components:**
  - ***If a model is to be useful it is necessary that anyone viewing the model can easily understand what it is saying***
- **Emphasize the main point :**
  - ***A model visualization should emphasize the main points and interactions***

NetLogo Ethnocentrism model



This model uses the shape of agents to visualize agent strategy

Batch vs. Interactive

- When user can interact as the model is running This kind of spur-of-the-moment control is called interactive running
- In contrast to interactive running, *another type of user running of a model is called batch running* With batch running, instead of controlling the model directly, *the user writes a script to run the model many times*

Summary

- When designing the user interface for your model, it is important to keep in mind that some users are going to want to be able to interact with your model
- Credits : Uri Wilensky

Modeling and Simulation

Schedule

Overview

- In NetLogo, there is no single identifiable object that can be identified as “ the schedule. ”
- We will first discuss the common SETUP/GO idiom which is employed in almost all agent-based models, and then move on to discuss some of the subtler issues concerning scheduling in ABMs.

Schedule, SETUP and GO

- The schedule is a description of the order in which the model operates
- There is usually an initialization procedure that creates the agents initializes the environment, and readies the user interface

SETUP and GO

- In NetLogo initialization procedure is usually called *SETUP* and it executes whenever a user presses the SETUP button on a NetLogo model
- For instance, in Traffic Basic the SETUP procedure looks

```
to setup
    clear-all
    ask patches [ setup-road ]
    setup-cars
    watch sample-car
end
```

Main Loop

```
to go
;; if there is a car right ahead of you, slow down to a speed below its speed
    ask turtles [
         let car-ahead one-of turtles-on patch-ahead 1
         ifelse car-ahead != nobody
              [   slow-down-car car-ahead]
              ;; otherwise, speed up
              [ speed-up-car ]
    ;; don't slow down below speed minimum or speed up beyond speed limit
         if speed < speed-min [ set speed speed-min ]
         if speed > speed-limit [ set speed speed-limit ]
         fd speed ]
    tick
end
```

The other main part of the schedule is what is often called the main loop, or in NetLogo, the GO procedure.

Traffic Basic the GO procedure looks like above code:

Asynchronous vs. Synchronous Updates

- If a model uses an *Asynchronous* update schedule,
  - This means that when agents change their state, that state is immediately seen by other agents.
- In a *Synchronous* update schedule changes made to an agent are not seen by other agents until the next clock tick
  - That is, all agents update simultaneously.

Sequential vs. Parallel Actions

- **Sequential actions involve only one agent acting at a time while Parallel actions are those in which all agents act independently.**
- **In NetLogo sequential action is the standard behavior for agents.**

Sequential vs. Parallel Actions

- For the agents to act truly in parallel, you would need parallel hardware so that the actions of each agent would be carried out by a separate processor.
- There is an intermediate solution.
  - *Simulated concurrency* uses one processor to simulate many agents acting in parallel

Summary

- After completing these Modules, you have learned how to use simple agent-based models, extend an ABM, and build your own agent-based model, while familiarizing yourself with the components of an agent-based model.
- Credits : Uri Wilensky

Modeling and Simulation

Analyzing Agent-Based Models

Overview

- **Here, we will understand details on how to analyze Agent-based Models.**

Analyzing Agent-Based Models

- We have examined, modified, and built models from scratch.
- Question: How can we employ ABMs to produce new and exciting results about the domain?

Analyzing Agent-Based Models

- **Specifically:**
- **What kind of results are produced by an ABM?**
- **Advantages and Disadvantages of comparative techniques**
- **Often times, it is better to consider the analysis method *before* building an ABM**

Analyzing Agent-Based Models

- Let us examine a disease spread model
- Someone catches a cold and is coughing up a storm, he might infect others.
- Those that he comes into contact with — his friends, co-workers, and even strangers — may catch the cold.

Analyzing Agent-Based Models

- We have examined, modified, and built models from scratch.
- Question: How can we employ ABMs to produce new and exciting results about the domain?

Analyzing Agent-Based Models

Infection Data

Infection Data

Population	50	100	150	200
Time to 100% Infection	419	188	169	127

Analyzing Agent-Based Models

- **Based on these results, we conclude that as the population density increases, the time to full infection dramatically decreases.**

Analyzing Agent-Based Models

Table 6.2
Your Friend's Data

Population	50	100	150	200
Time to 100% Infection	305	263	118	126

Table 6.3
Raw Data

Population	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
50	419	365	305	318	323	337	432	380	430	359
100	188	263	256	205	206	205	201	181	202	231
150	169	118	163	146	143	167	137	121	140	140
200	127	126	113	111	133	129	109	101	105	133

Analyzing Agent-Based Models

- The data is inconsistent because most ABM models employ randomness in their algorithms
- i.e., the code makes use of a random number generator.

Summary

- We have learnt about the analysis of agent-based models.
- Credits: Uri Wilensky book.

Modeling and Simulation

Types of Measurements

Overview

- **Here, we will understand about the different types of measurements in an ABM**

Types of Measurements

- **Statistical Analysis of ABM: Moving beyond Raw Data**
- **Statistical results are the most common way of looking at any kind of scientific data**

Types of Measurements

- The general methodology behind descriptive statistics is to provide numerical measures
- These measures summarize a large data set
- Also, describe the data set in such a way that it is not necessary to examine every single value.

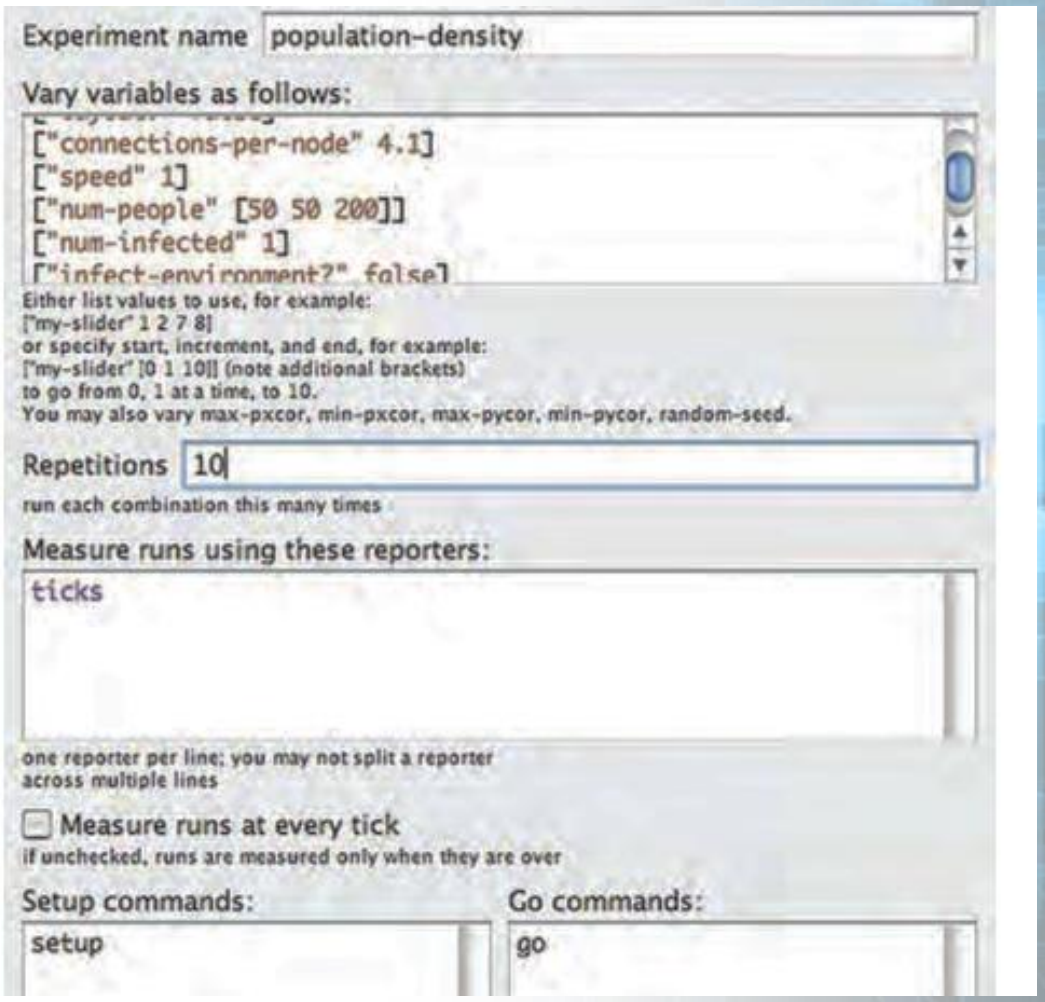
Types of Measurements

Summary Statistics

Population	Mean	Std. Dev.
50	366.8	47.39385802
100	213.8	27.40154091
150	144.4	17.65219533
200	118.7	12.12939497

Types of Measurements

Behavior space experiments



Types of Measurements

BehaviorSpace Data Imported into a Spreadsheet

BehaviorSpace Table data

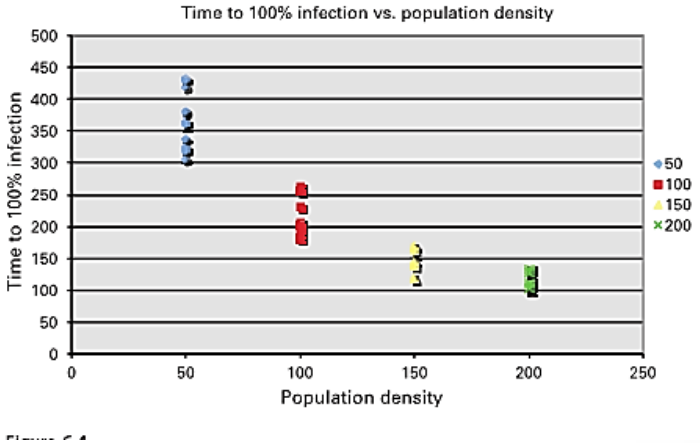
population-density

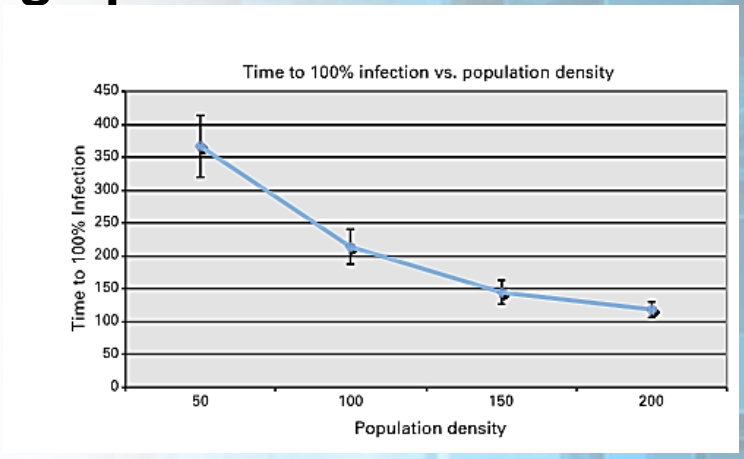
DATE
TIME

[run number]	network?	layout?	connections-per-node	speed	num-people	num-infected	infect-environment?	[tick]	ticks
1	FALSE	FALSE	4.1	1	50	1	FALSE	299	299
2	FALSE	FALSE	4.1	1	50	1	FALSE	432	432
3	FALSE	FALSE	4.1	1	50	1	FALSE	444	444
4	FALSE	FALSE	4.1	1	50	1	FALSE	400	400
5	FALSE	FALSE	4.1	1	50	1	FALSE	467	467
6	FALSE	FALSE	4.1	1	50	1	FALSE	397	397
7	FALSE	FALSE	4.1	1	50	1	FALSE	337	337

Types of Measurements

- **Raw data in graphs**





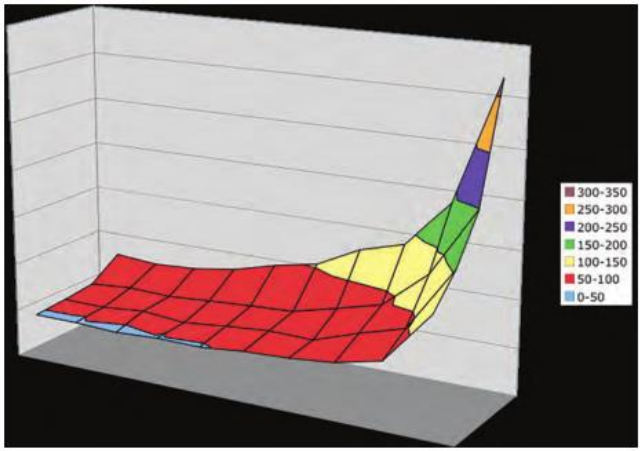


Figure 6.9

3D Chart of NUM-PEOPLE and DISEASE-DECAY versus time to 100 percent infection.

Summary

- We have learnt about the types of measurements in the analysis of an ABM.
- Credits: Uri Wilensky book.

Modeling and Simulation

Modeling the Spread of Disease

Overview

- **Here we will Model the spread of disease in detail**

Infection Initially Rises Exponentially

- If a cold virus infects someone, that person might spread that disease to five other people (six now infected) before they recover.
- In fact, the rate of infection initially rises exponentially.

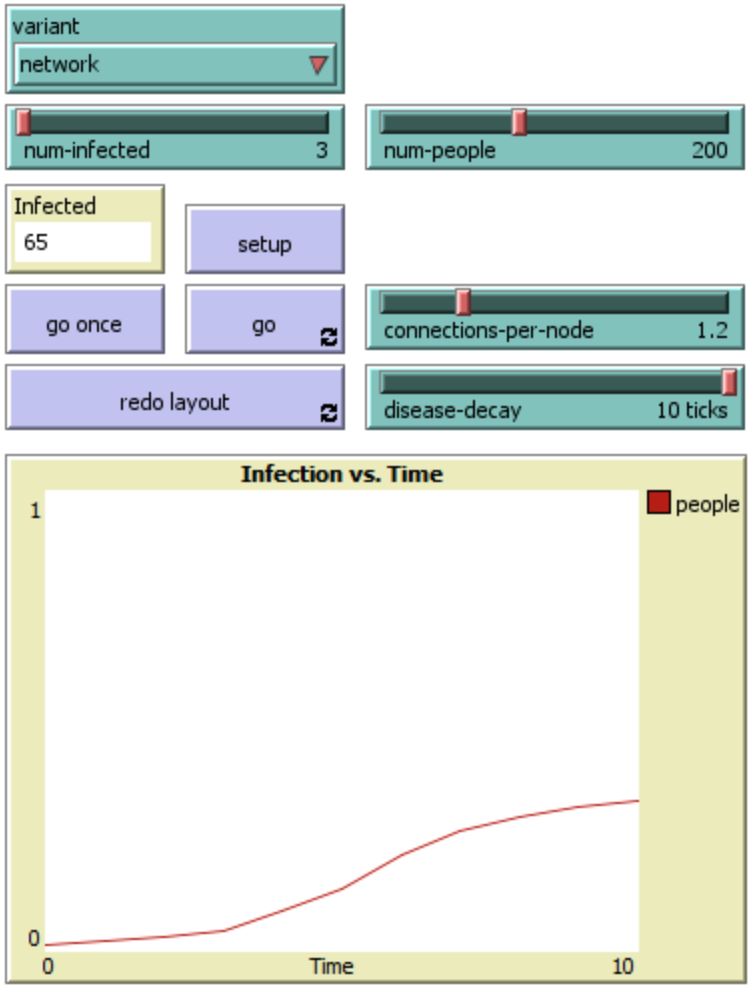
Understanding the Spread of Disease

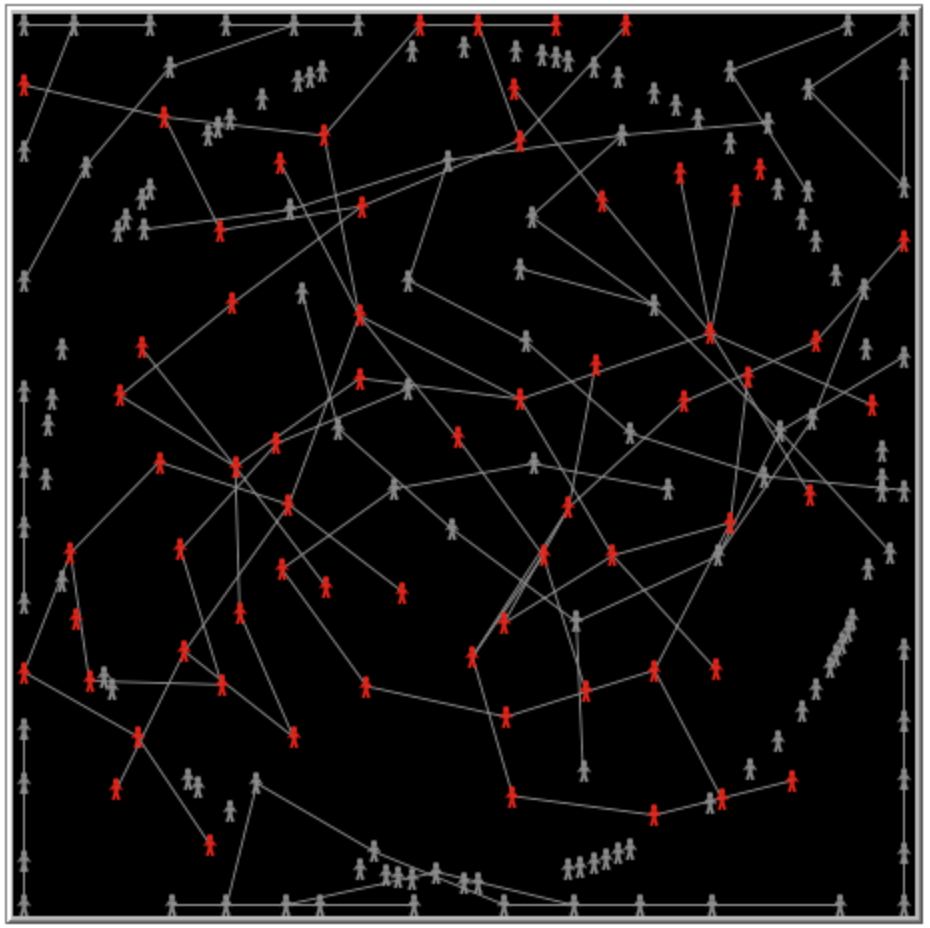
- **Suppose that we are interested in understanding the spread of disease, and we want to build an ABM of such a spread.** ***How should we go about doing it?***
  - ***First, we need some agents that keep track of whether they are infected with a cold or not***

Initialize the Model

- We need the ability to initialize the model by infecting a group of individuals
- Individuals move around randomly on a landscape and infect other individuals whenever they come into contact with them.

Spread of Disease model





We conclude that as the population density increases, the time to full infection dramatically decreases

In Beginning infection rate increases slowly

- In the beginning, when *the first person becomes infected, if there are not many other people around, the person has no one to infect*, and thus the infection rate increases slowly.
- However, if there are many people around then there will be plenty of infection opportunities

True Despite the Fact

- **Moreover, at the end of the run, when there are only one or two uninfected agents, they will be more likely to run into someone with an infection if the population count is high.**
- ***This is true despite the fact that the total number of people that need to be infected increases.***

Summary

- To describe these patterns of behavior it makes sense to turn to some statistics
- Credits : Uri Wilensky

Modeling and Simulation

Statistical Analysis of ABM: Moving beyond Raw Data

Overview

- **Here we will learn about the general methodology behind descriptive statistics that is to provide numerical measures that summarize a large data set and describe the data set in such a way that it is not necessary to examine every single value**

A Coin Is Fair

- **Suppose we are interested in determining whether a coin is fair**
  - **(i.e., it is as likely to turn up heads when flipped as it is to turn up tails) then we can conduct a series of experiments where we flip the coin and observe the results.**

Means And Standard Deviations

- It is much easier to look at means and standard deviations than it is to examine large series of data
  - (e.g., for HHHHTTHTTT, the observed probability is 0.5, and the expected outcome for ten trials is to observe five heads with a standard deviation of 1.58).

Spread of Disease model

- To apply this technique to our Spread of Disease model in more depth, we can create summary statistics

Summary Statistics

Population	Mean	Std. Dev.
50	366.8	47.39385802
100	213.8	27.40154091
150	144.4	17.65219533
200	118.7	12.12939497

- From these summary results:

Summary Statistics

- We see that the mean time to 100 percent infection declines as the population density increases.
- Another interesting result is that as the population density goes up, the standard deviation goes down.
- This means that the data is less varied.

Confirming Or Rejecting Hypotheses

- These results seem to confirm our original hypothesis that as population density increases the mean time to infection declines.
- Within ABM, statistical analysis is a common method of confirming or rejecting hypotheses

Abms Create Large Amounts Of Data

- **ABMs create large amounts of data (the Spread of Disease model is just a small example), and if we can summarize that data we can examine large amounts of output in an efficient manner.**

ABM toolkits

- Most ABM toolkits give you the basic ability to carry out simple statistical analysis within the package itself
  - (e.g., in NetLogo there are MEAN and STANDARD-DEVIATION primitives). Thus, while the model is running, the ABM itself can generate summary statistics.

Summary

- The NetLogo R extension to conduct analyses with the R statistical package
- Credits : Uri Wilensky

Modeling and Simulation

The Necessity of Multiple Runs within ABM

Overview

- When you are trying to collect statistical results from an ABM you should run the model multiple times and collect different results at different points
- Here we will understand how ABM toolkits will provide you with a way to collect the data from these runs automatically

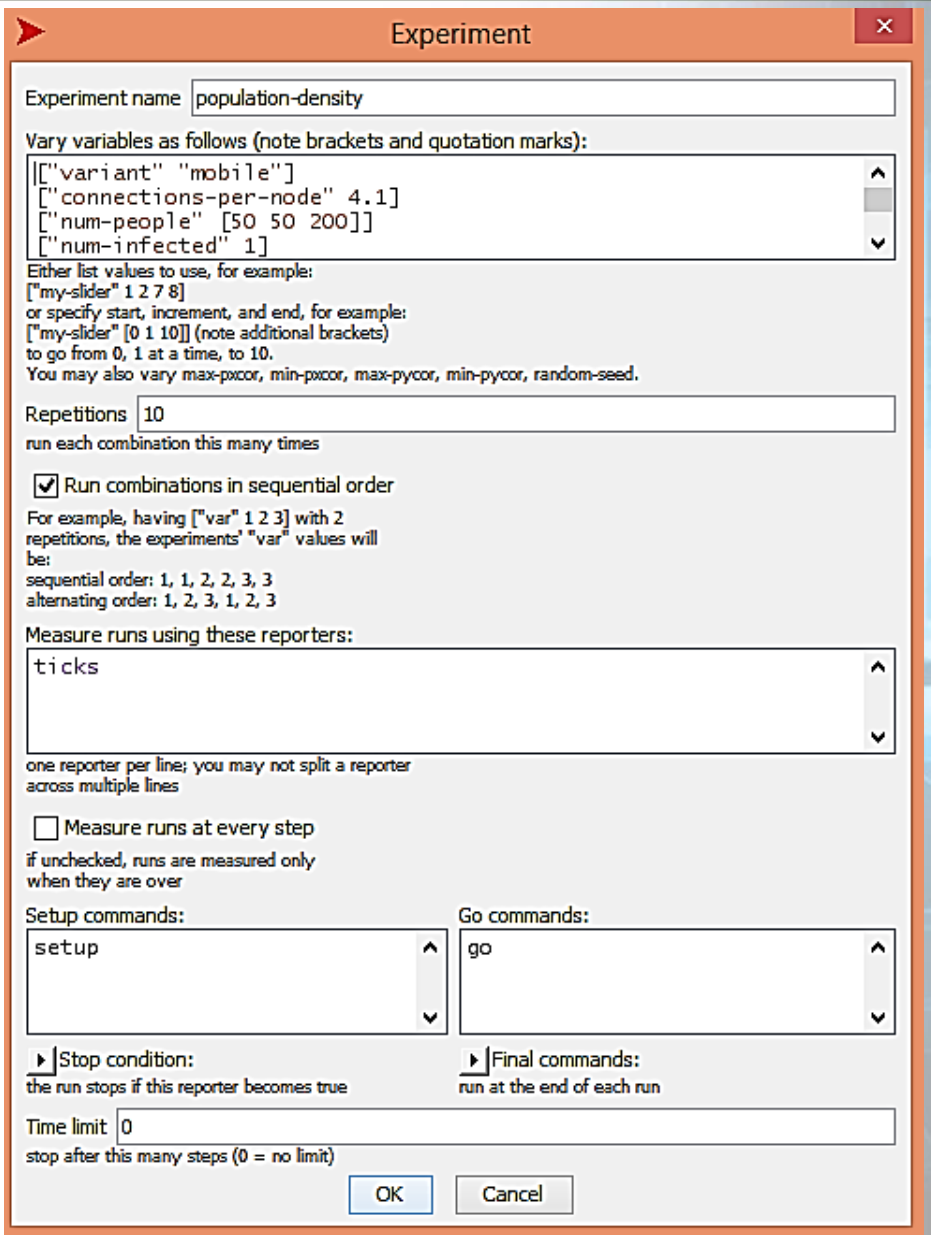
BehaviorSpace

- In NetLogo there is a tool called BehaviorSpace
- ABM toolkits are often full-featured programming languages, allowing you to write your own tools for creating experiments to produce the data sets you want to analyze

Batch Experiment Tool

- These tools will automatically run the model multiple times with multiple different settings and collect the results in some easy to use format like the CSV files mentioned earlier.
- Let us call this experiment “ population density, ”

Setting Up a BehaviorSpace Experiment



Commands

- The SETUP and GO commands allow you to specify any additional NetLogo code that you need to make the model start and go.
  - “ Stop condition ” allows you to specify special stop conditions for each run, and
  - “ Final commands ” allows you to insert any commands that you want executed between runs of the model

Manually Running Model Multiple

- **In general in ABM, it is important to carry out multiple runs of your experiments so that you can determine if some result is truly a pattern or just a one-time occurrence.**

Summary

- One common way is to start by manually running your model multiple times, but to get a better sense of the results it is usually much easier to use a batch experiment tool We have illustrated the BehaviorSpace tool, which is the batch experiment tool for NetLogo
- Credits : Uri Wilensky

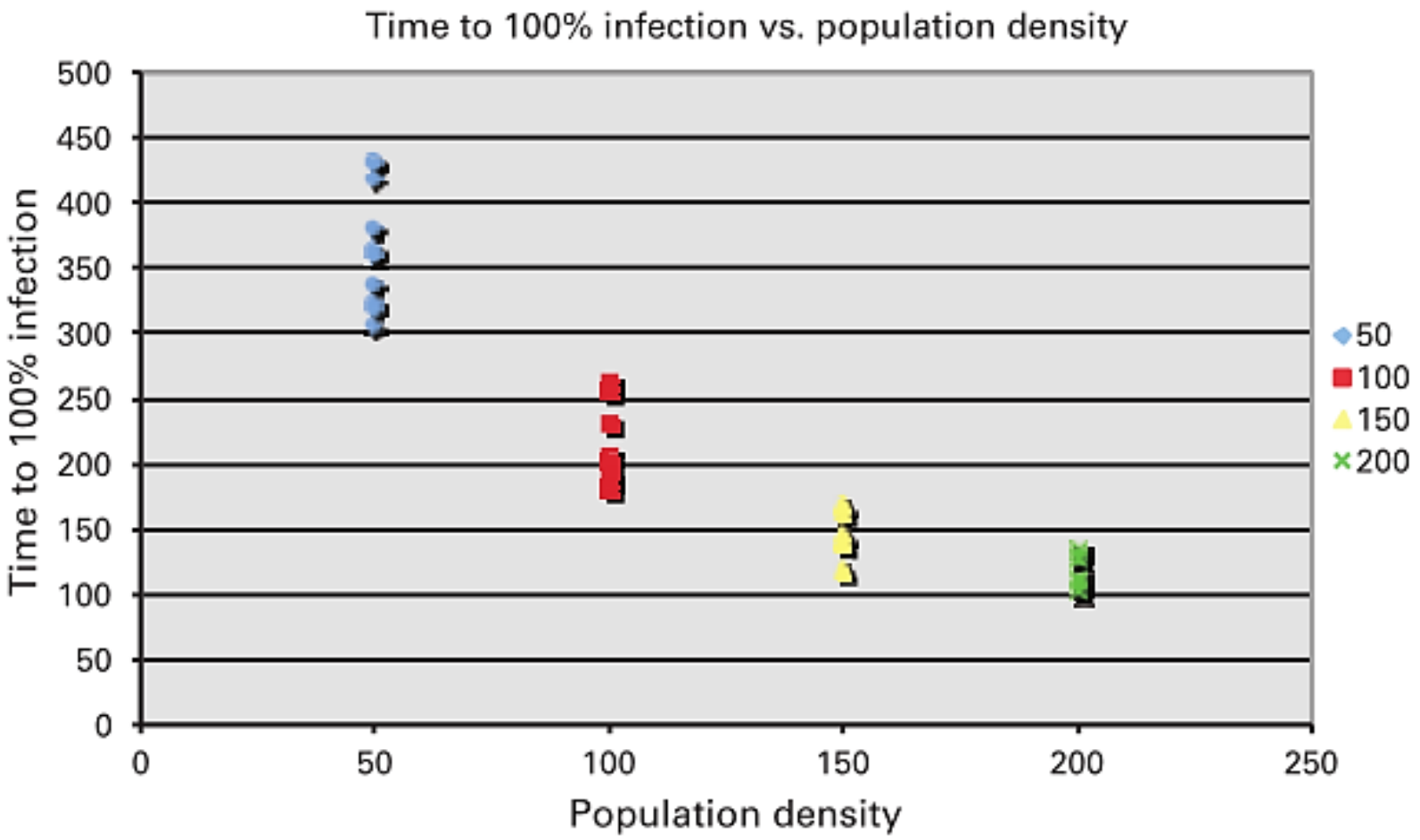
Modeling and Simulation

Using Graphs to Examine Results in ABM

Overview

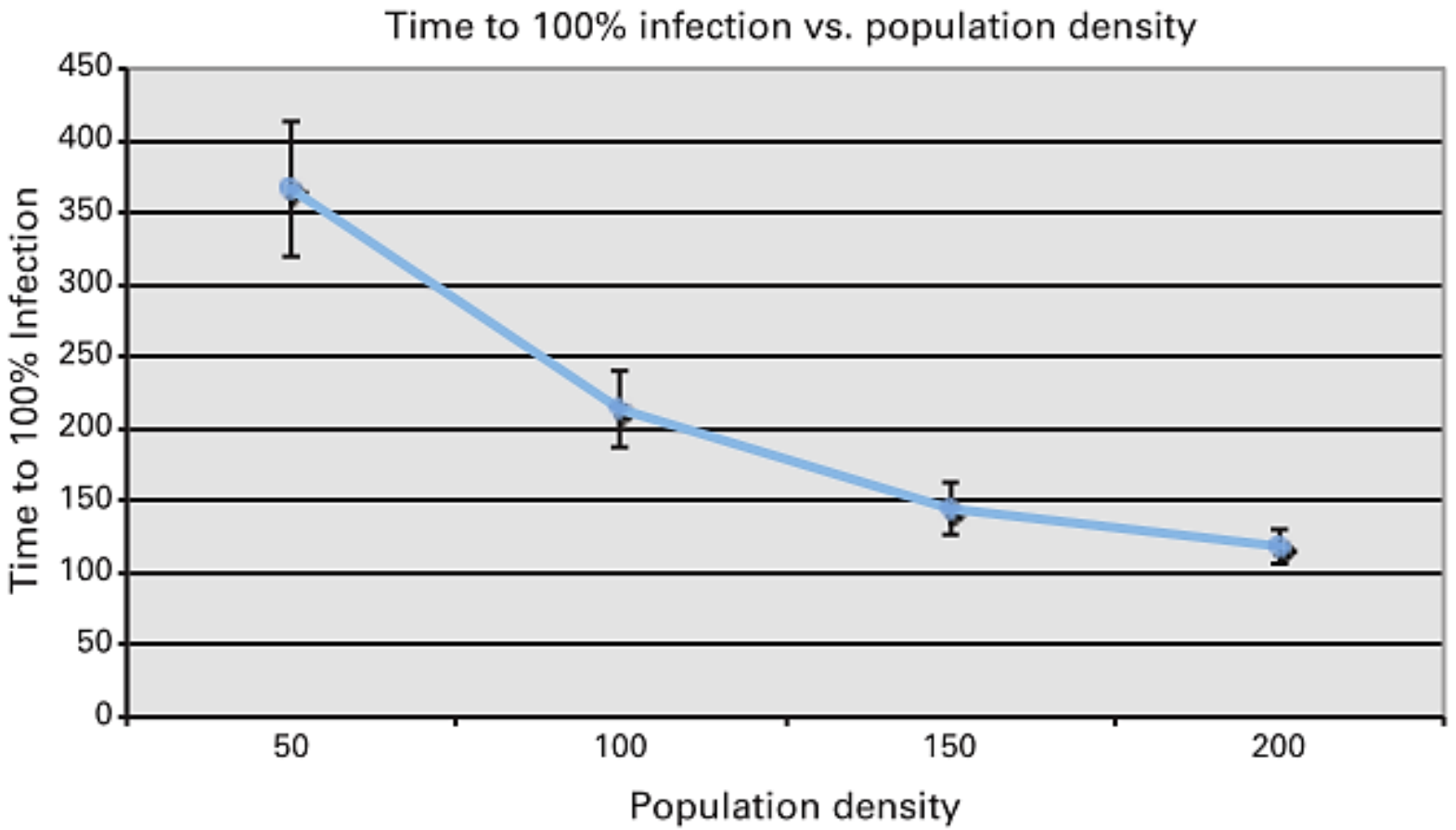
- In NetLogo creating simple graphs is easy to do and will often suffice for simple data analysis.
- But with complex data sets, designing a useful and immediately informative graph can be challenging and is the subject of an extensive body of literature

All Raw Data In a Graph-based Form



We can quickly see how the data is distributed and how the data changes with population density

Summary data

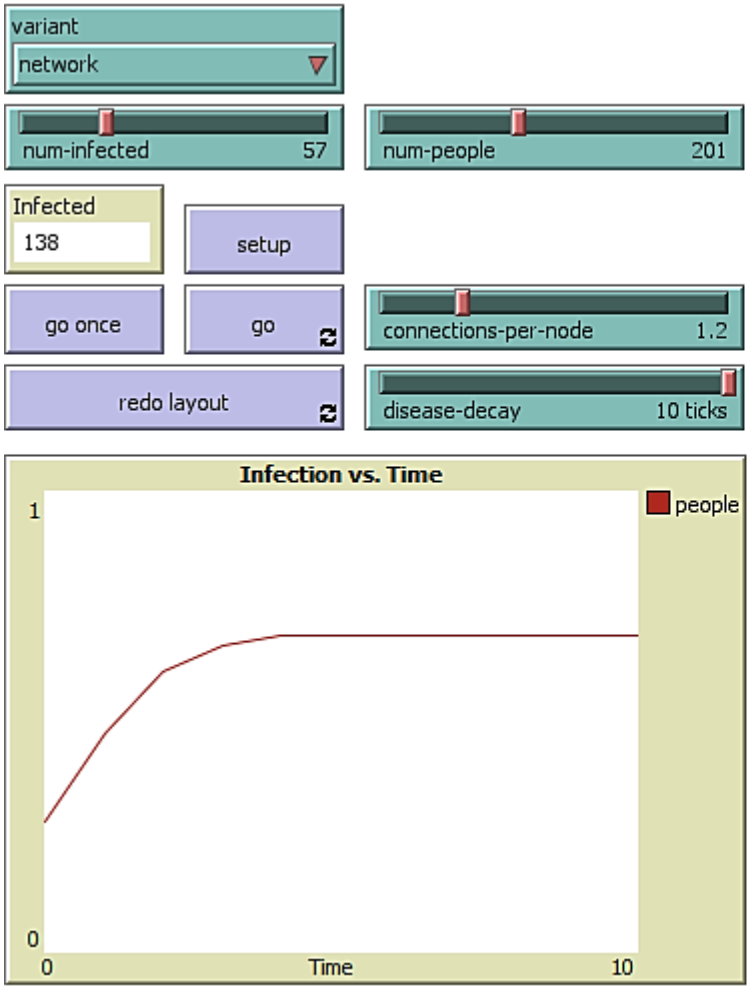


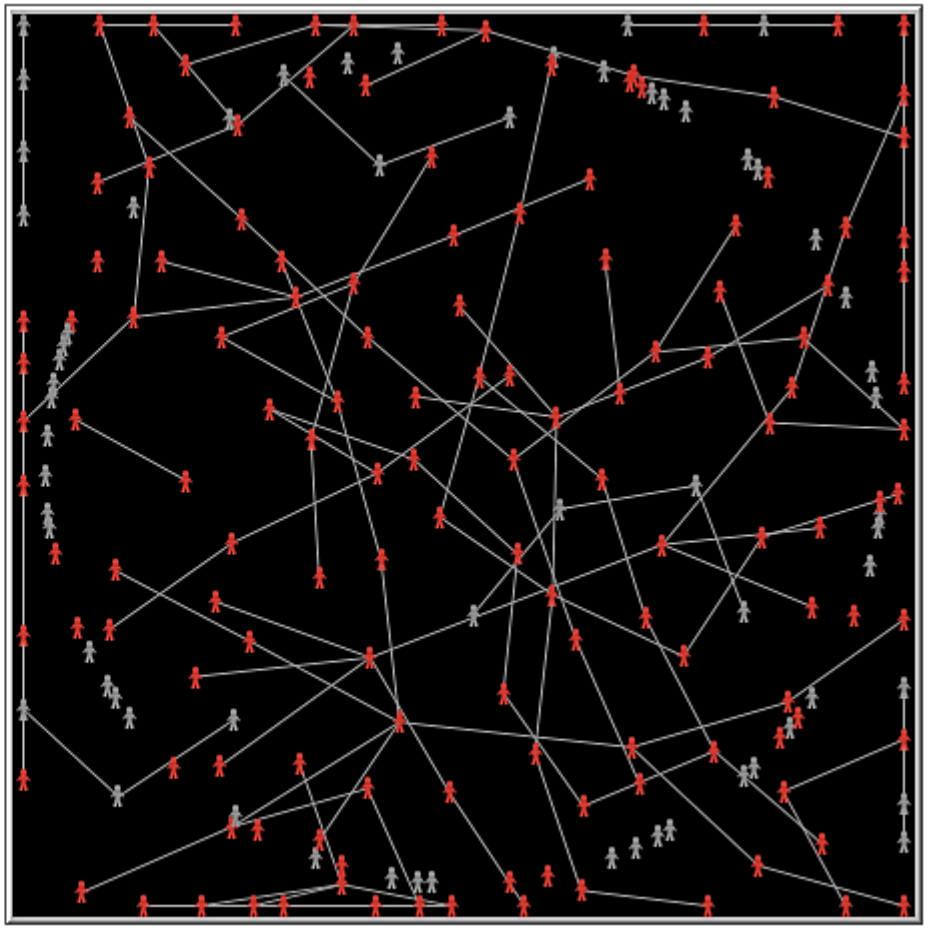
This new figure might not be easier to understand than previous figure, but if there were one hundred data points in previous figure rather than ten data points, then a figure like this one might be very helpful.

Graphs

- Many ABM toolkits *include capabilities for continually updating graphs and charts* during the running of a model and thus enable you to see the progress of the model temporally
- For instance, in the Spread of Disease model there is a graph that *illustrates the change in the fraction of infected agents as time proceeds*

Fraction Infected Versus Time





This is one example of how we can use time series to help understand the behavior of a model.

Summary

- Summary: We have seen the use of graphs in simulation software
- Credits: Uri Wilensky

Modeling and Simulation

Analyzing Networks within ABM

Overview

- **Here, we understand the analysis of “networks within ABM”**

Spread of disease model

- Interactions do not occur in physical space.
- But rather across social networks.

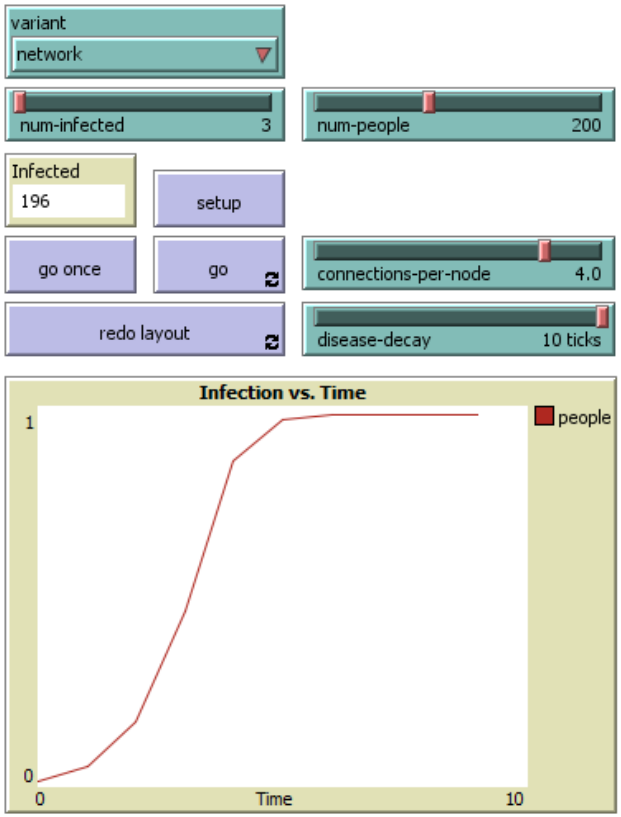
Spread of disease model

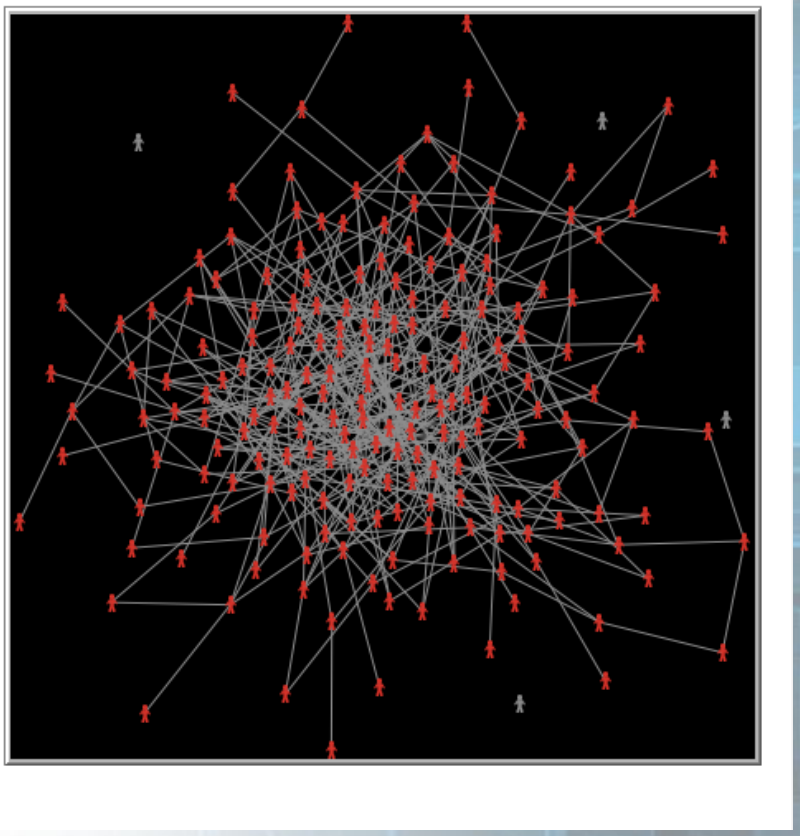
- Some diseases spread only through certain kinds of social networks.
- If we set the chooser to “network” we can explore this.

Spread of disease model

- The property which determines how many connections.
- Each individual has to other individual in the network.
- Known as “connections-per-node”.

Spread of disease model figure





Spread of disease model

- A well known property of random graph is the average number of individuals infected.
- Grows substantially
- The connections-per-node exceeds 1.0
- Forms a giant component in the network.

Spread of disease model

- The property of average path length, which measure the distance between any two nodes in the network.
- Affect the spread of disease in the network.
- Another property is clustering co-efficient.

Spread of disease model

- Different properties and tools are there to measure and analyze a wide variety of metrices
- Associated with social network analysis (SNA).

Summary

- Each of these network properties can be analyzed as to their effect on the spread of disease.
- Reports and toolkits like UCInet can be used for further examination.
- Credits: Uri Wilensky book.

Modeling and Simulation

Environmental Data and ABM

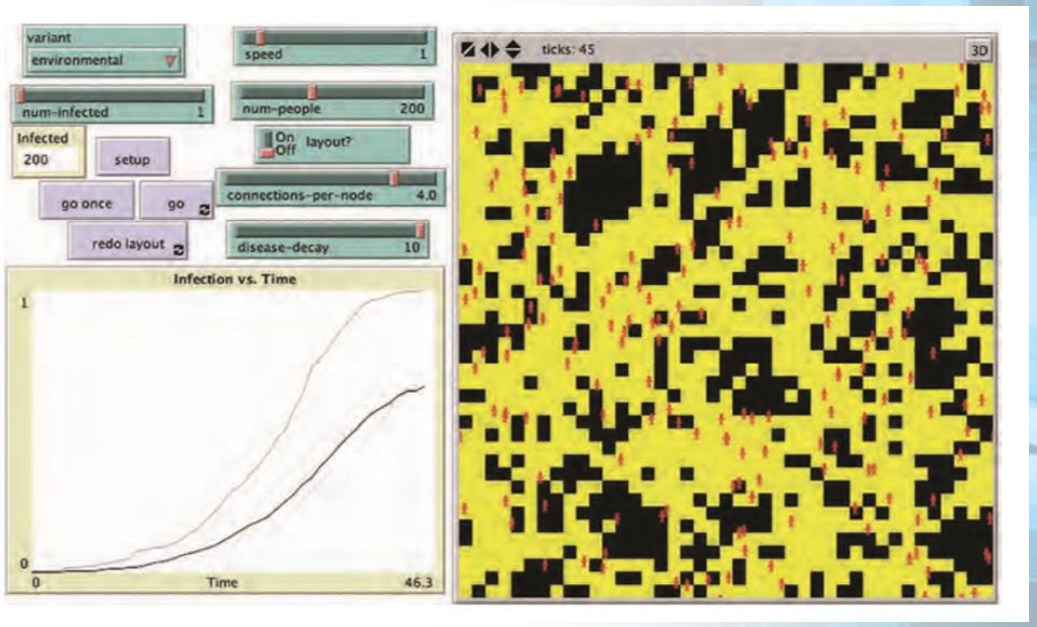
Overview

- **In this section we are going to see agent-to-environment and environment-to-agent transmission.**

Environmental data and ABMs

- Spread of disease model for examining.
- Environmental interaction effect.

Environmental data and ABMs



Environmental data and ABMs

- The path below any agent will become yellow.
- Change the rate if DISEASE-DECAY 0 to 10 at a single time intervals

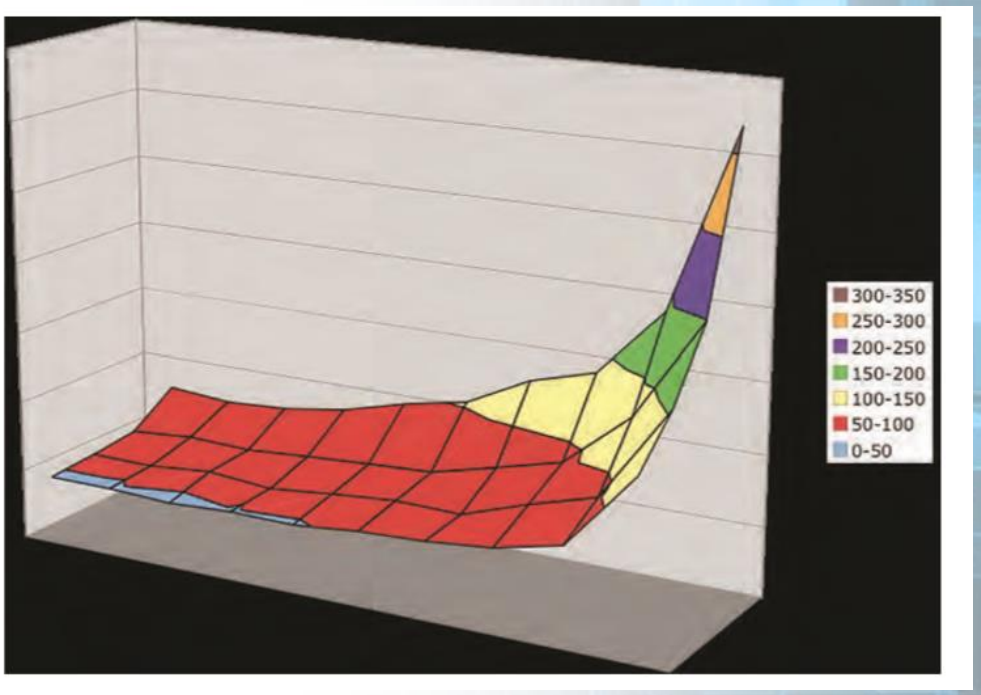
Environmental data and ABMs

- A long DIESEASE-DECAY might have negligible over having a small DISEASE-DECAY.
- Investigating using behaviour space.

Environmental data and ABMs

- **A Powerful aspect of ABMs is that it also shows us the pattern of infection.**

Environmental data and ABMs



Environmental data and ABMs

- **Leaving long stringy patterns of environmental infection.**

Summary

- We have seen the working of environmental data and ABMs.
- Credits: Uri Wilensky book

Modeling and Simulation

Correctness of a Model

Overview

- **Here, we understand the about the “correctness of a model”**

Correctness of a Model

- Output relating the concerned issue must be accurate.
- Model accuracy is evaluated through validation, verification and replication.

Correctness of a Model

- **Implemented model corresponds to, and explains, some phenomenon in the real world.**

Correctness of a Model

- **Model verification: determining whether an implemented model corresponds to the target conceptual model.**
- **Make sure that the model has been implemented correctly.**

Correctness of a Model

- **Model replication is the implementation by one researcher or group of researchers of a conceptual model previously implemented by someone else.**

Correctness of a Model

- Set of results from a model that corresponds to the real world is not sufficient.
- Multiple runs are often needed to confirm that a model is accurate.

Summary

- **Verification, validation, and replication collectively underpin the correctness, and thus utility, of a model.**
- **Credits: Uri Wilensky book.**

Modeling and Simulation

Verification

Overview

- **Here, we understand the role of “Verification” in Correctness of a Model.**

Verification

- In larger models, the code can be difficult to understand as it evolves over time.
- Verification ensures the elimination of “bugs” from the code.

Verification

- The process of debugging becomes more difficult for complex models.
- Keep the process easy and simple.

Verification

- **Build the model simply to begin with.**
- **It will be easier to verify.**

Verification

- Expand the complexity of the model as necessary.
- This incremental approach makes it easy to verify the additional components.

Verification

- Even if all the components are verified, it is still possible that the system isn't.
- Complications may arise from the interaction between model components.

Summary

- Throughout the section , we examined the issue of verification of a model in the context of a simple ABM.
- Credits: Uri Wilensky book

Modeling and Simulation

Communication

Overview

- **Here, we understand need of “communication” for Correctness of Model.**

Communication

- Sometimes a team of people builds a model.
- Other team members actually implement the model.
- Verification becomes critical.

Communication

- **Communication is critical to ensure that the implemented model is correct.**
- **It is essential.**

Communication

- In the voting model.
- Political scientists differentiate between Moore and Van Neumann neighborhoods.
- Small world network and hexagonal versus a rectangular grid.

Communication

- In an ideal situation, the model author and the implementer is the same person which averts the sort of communication errors.
- But when it isn't then there is often room for human error and misunderstanding.

Communication

- In the past it was difficult to be expert model implementer and model author.
- However, low-threshold ABM languages, such as NetLogo is narrowing the gap between author and implementer.

Summary

- We have seen how communication play an important role in the correctness of a model.
- How communication can fill the gap between the implementor and the author of the model.
- Credits: Uri Wilensky book.

Modeling and Simulation

Describing Conceptual Models

Describing Conceptual Models

Overview

- **Here, we will understand how "Describing Conceptual Model" play a role in correctness of a model.**

Describing Conceptual Models

Conceptual Models

- **Implementing the voting model.**
- **We may realize that we didn’t understand completely when we talked to political scientist.**

Describing Conceptual Models

Conceptual Models

- Describing how we plan to implement the model.
- We and political scientist have same conceptual model in mind.
- This document is “Describing conceptual model”.

Describing Conceptual Models

Conceptual Models

- **Describe the model in more formal terms.**
- **Pictorial description of the model using flowcharts.**

Describing Conceptual Models

Conceptual Models

- **Convert the flowcharts into pseudo-code**
- **The goal of pseudo-code is to serve as a midway point between natural language and programming language.**

Describing Conceptual Models

Pseudo-Code

```
Voters have votes = {0, 1}

For each voter:
    Set vote either 0 or 1, chosen with equal probability
Loop until election
    For each voter
       If majority of neighbors' votes = 1 and vote = 0 then set vote 1
       Else If majority of neighbors' votes = 0 and vote = 1 then set vote 0
       If vote = 1: set color blue
       Else: color = green
    Display count of voters with vote = 1
    Display count of voters with vote = 0
End loop
```

Describing Conceptual Models

Conceptual Models

- **Other methods are UML, choosing a language similar to pseudo-code and NetLogo.**

Describing Conceptual Models

Summary

- We have learnt about processes involve in describing conceptual model which include flowcharts, pseudo-code, UML and NetLogo.
- Credits: Uri Wilensky book

Modeling and Simulation

Verification Testing

Verification Testing

Overview

- **Here, We will discuss and understand how we can use “Verification Testing” for the Correctness of a Model.**

Verification Testing

Implementation

- **When implementing the design into code we follow ABM core design principle.**

Verification Testing

Setup Code

```
patches-own
[
    vote ;; my vote (0 or 1)
    total ;; sum of votes around me
]

to setup
    clear-all
    ask patches [
           if (random 2 = 0) ;; half a chance of this
           [ set vote 1 ]
       ]
    ask patches [
       if (random 2 = 0) ;; half a chance of this
           [ set vote 0 ]
    ]
    ask patches [
       recolor-patch
    ]
end

to recolor-patch ;; patch procedure
    ifelse vote = 0
       [ set pcolor green ]
       [ set pcolor blue ]
end
```

Verification Testing

To Check Setup

```
to check-setup
    let diff abs ( count patches with [ vote = 0 ] - count patches
with [ vote = 1 ] )
    if diff > .1 * count patches [
       print "Warning: Difference in initial voters is greater than 10^4%."
    ]
end
```

Verification Testing

To Check Setup

```
to setup
    clear-all
    ask patches [
           set vote random 2     ;; 0 or 1, with equal probability
    ]
    ask patches [
       recolor-patch
    ]
    check-setup
end
```

Verification Testing

Unit Testing

- This verification technique is a form of unit testing.
- We can modify the code without disrupting previous code.

Verification Testing

NetLogo Code

```
to go
    ask patches [
       set total (sum [vote] of neighbors)
    ]
    ;; this is equivalent to count neighbors with [vote = 1]
    ;; use two ask patches blocks so all patches compute "total"
    ;; before any patches change their votes
    ask patches [
       ifelse vote = 0 and total >= 4 [
           set vote 1
       ]
           [if vote = 1 and total <= 4 ] [
               [set vote 0
           ]
           recolor-patch
       ]
       tick
       end
```

Verification Testing

Summary

- We understand how incremental approach makes it easy to implement to model correctly.
- Then we write our code into NetLogo for verification purpose.
- Credits: Uri Wilensky book

Modeling and Simulation

Beyond Verification

Beyond Verification

Overview

- **Here, we will learn about the beyond verification using agent-based modelling.**

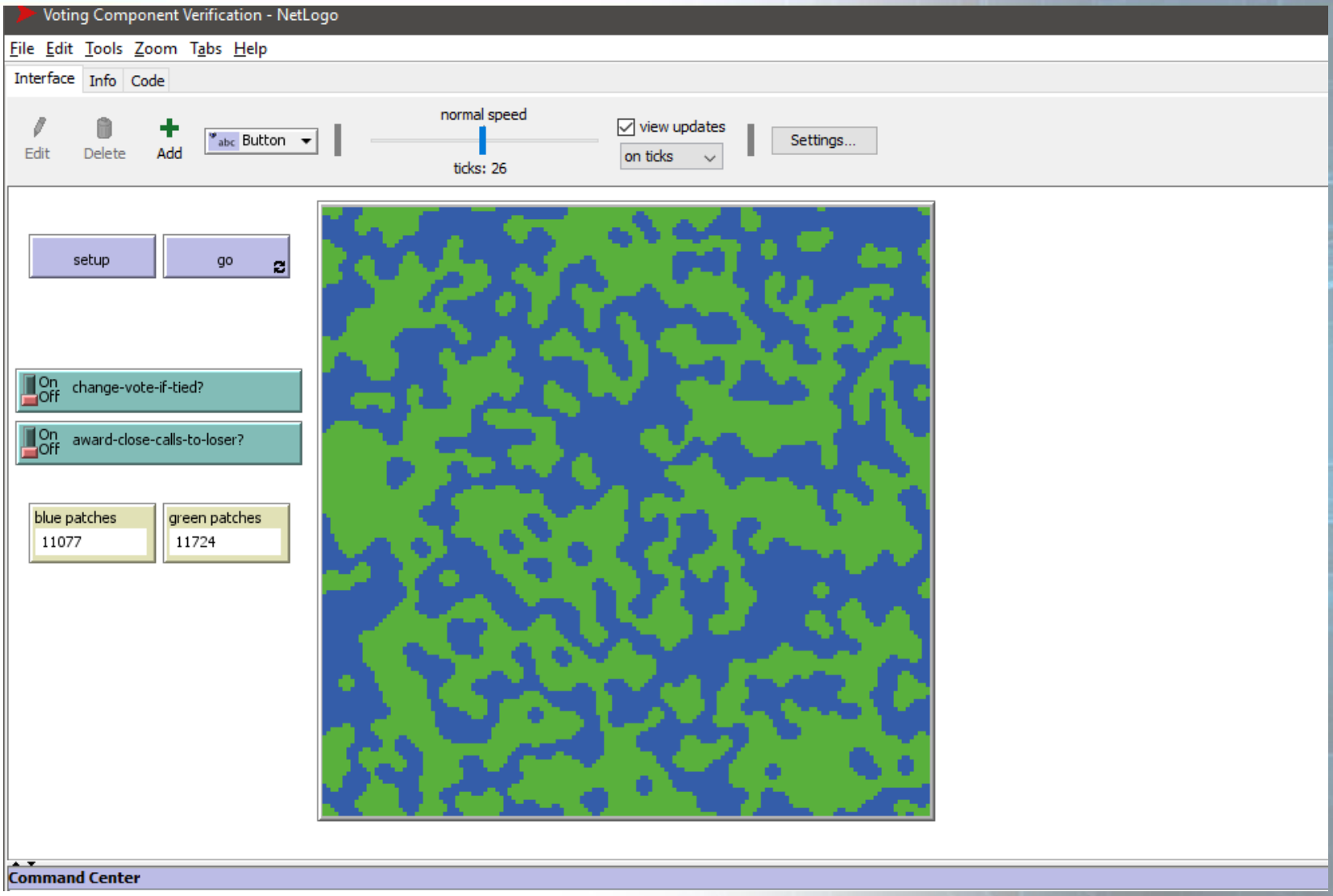
Beyond Verification

- Sometimes results are not produced to what the implementers and authors hypothesized.
- Results of modelling confuse the scientists.

Beyond Verification

- Jagged edges due to tie votes.
- Design the model that neighbours do not change their votes if tied.
- Switch called CHANGE VOTE IF TIED.

Beyond Verification



Beyond Verification

```
patches-own [
  vote   ;; my vote (0 or 1)
  total  ;; sum of votes around me
]

to setup
  clear-all
  ask patches [
    set vote random 2 ;; set vote to either 0 or 1
    recolor-patch
  ]
  reset-ticks
  check-setup
end

to go
  ;; keep track of whether any patch has changed their vote
  let any-votes-changed? false
  ask patches [
    set total (sum [ vote ] of neighbors)
  ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
  ask patches [
    let previous-vote vote
    if total < 3 [ set vote 0 ] ;; if majority of your neighbors vote 0, set your vote to 0
    if total = 3 [
      ifelse award-close-calls-to-loser?
```

Beyond Verification

```
        [ set vote 1 ]
        [ set vote 0 ]
    ]
    if total = 4 and change-vote-if-tied? [
      set vote (1 - vote) ;; invert the vote
    ]
    if total = 5 [
      ifelse award-close-calls-to-loser?
        [ set vote 0 ]
        [ set vote 1 ]
    ]
    if total > 5 [ set vote 1 ] ;; if majority of your neighbors vote 1, set your vote to 1
    if vote != previous-vote [ set any-votes-changed? true ]
    recolor-patch
  ]
  ;; if the votes have stabilized, we stop the simulation
  if not any-votes-changed? [ stop ]
  tick
end

to recolor-patch  ;; patch procedure
  ifelse vote = 0
    [ set pcolor green ]
    [ set pcolor blue ]
end

;; This procedure checks to see if the SETUP procedure sets up the model with roughly
;; equal numbers of blue and green patches
to check-setup
  ;; count the difference between the number of green and the number of blue patches
  let diff abs (count patches with [ vote = 0 ] - count patches with [ vote = 1 ])
  if diff > .1 * count patches [
    print "Warning: Difference in initial voters is greater than 10%."
  ]
end


; Copyright 2008 Uri Wilensky.
```

Beyond Verification

- **Switch AWARD CLOSE CALLS TO LOSER?**
- **With both switches on, we have a different outcome.**

Beyond Verification

```
to go
  ask patches
    [ set total (sum [vote] of neighbors) ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
  ask patches
    [ if total > 5 [ set vote 1 ]
      if total < 3 [ set vote 0 ]
      if total = 4
        [ if change-vote-if-tied?
          [ set vote (1 - vote) ] ] ;; switch vote
      if total = 5
        [ ifelse award-close-calls-to-loser?
          [ set vote 0 ]
          [ set vote 1 ] ]
      if total = 3
        [ ifelse award-close-calls-to-loser?
          [ set vote 1 ]
          [ set vote 0 ] ]
      recolor-patch ]
  tick
end
```

Beyond Verification

summary

- We have seen the details of beyond verification.
- Credits: Uri Wilensky book

Modeling and Simulation

Sensitivity Analysis and Robustness

Sensitivity Analysis and Robustness

Overview

- **Here, we will understand the sensitive analysis and robustness**

Sensitivity Analysis and Robustness

- Sensitivity analysis.
- Creating the parameter to test the hypothesis of initial balance in one direction.
- Using BehaviourSpace running the experiment varying from 25 to 75 percent increment.

Sensitivity Analysis and Robustness

- Two conditions:
- If no voter switch vote in the last step the model will stop.
- The model will stop after one hundred times the steps have executed.

Sensitivity Analysis and Robustness

```
patches-own
[
  vote   ;; my vote (0 or 1)
  total  ;; sum of votes around me
]

to setup
  clear-all
  ask patches [
    ifelse random 100 < initial-green-pct
      [ set vote 0 ]
      [ set vote 1 ]
    recolor-patch
  ]
  reset-ticks
  check-setup
end

to go
  ;; keep track of whether any patch has changed their vote
  let any-votes-changed? false
  ask patches [
    set total (sum [ vote ] of neighbors)
  ]
  ;; use two ask patches blocks so all patches compute "total"
  ;; before any patches change their votes
```

Sensitivity Analysis and Robustness

```
]
;; use two ask patches blocks so all patches compute "total"
;; before any patches change their votes
ask patches [
  let previous-vote vote
  if total < 3 [ set vote 0 ] ;; if majority of your neighbors vote 0, set your vote to 0
  if total = 3 [
    ifelse award-close-calls-to-loser?
      [ set vote 1 ]
      [ set vote 0 ]
  ]
  if total = 4 and change-vote-if-tied? [
    set vote (1 - vote) ;; invert the vote
  ]
  if total = 5 [
    ifelse award-close-calls-to-loser?
      [ set vote 0 ]
      [ set vote 1 ]
  ]
  if total > 5 [ set vote 1 ] ;; if majority of your neighbors vote 1, set your vote to 1
  if vote != previous-vote [ set any-votes-changed? true ]
  recolor-patch
]
```

Sensitivity Analysis and Robustness

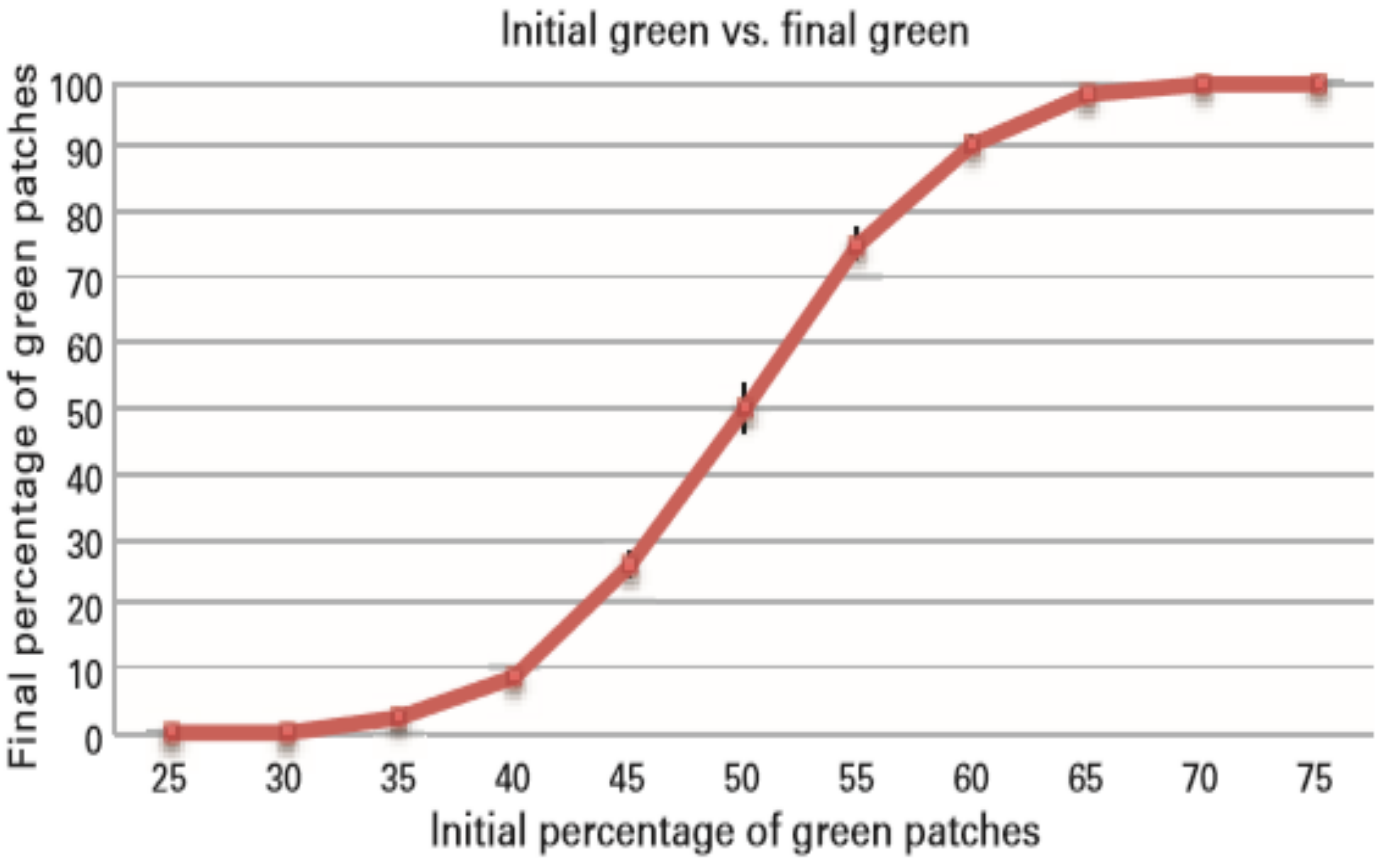
```
  ;; if the votes have stabilized, we stop the simulation
  if not any-votes-changed? [ stop ]
  tick
end

to recolor-patch  ;; patch procedure
  ifelse vote = 0
    [ set pcolor green ]
    [ set pcolor blue ]
end

;; This procedure checks to see if the SETUP procedure sets up the model with
;; roughly expected numbers, given the value of the initial-green-pct slider
to check-setup
  let expected-green (count patches * initial-green-pct / 100)
  let diff-green (count patches with [ vote = 0 ]) - expected-green
  if diff-green > (.1 * expected-green) [
    print "Initial number of green voters is more than expected."
  ]
  if diff-green < (- .1 * expected-green) [
    print "Initial number of green voters is less than expected."
  ]
end


; Copyright 2008 Uri Wilensky.
; See Info tab for full copyright and license.
```

Sensitivity Analysis and Robustness



Sensitivity Analysis and Robustness

- **Past research methodologies for sensitivity analysis.**
- **Active Nonlinear Testing.**

Sensitivity Analysis and Robustness

Summary

- We have seen the details of Sensitivity Analysis and Robustness
- Credits : Uri Wilensky book

Modeling and Simulation

Verification Benefits and Issues

Verification Benefits and Issues

Overview

- **Here, we are going to look verification benefits and issues.**

Verification Benefits and Issues

- Understanding the cause of unexpected outcomes.
- Impact of small changes.

Verification Benefits and Issues

- Implemented model corresponds to the conceptual model.
- Model's low level rules.
- Understanding the mechanisms.

Verification Benefits and Issues

- **A bug in the code can produce surprising results.**

Verification Benefits and Issues

- **Understanding the operating of the model.**
- **A verification process is not binary.**

Verification Benefits and Issues

Summary

- We have seen the verification benefits and issues.
- Credits: Uri Wilensky book

Modeling and Simulation

Validation

Validation

Overview

- **Here: we will see the validation of agent based modeling.**

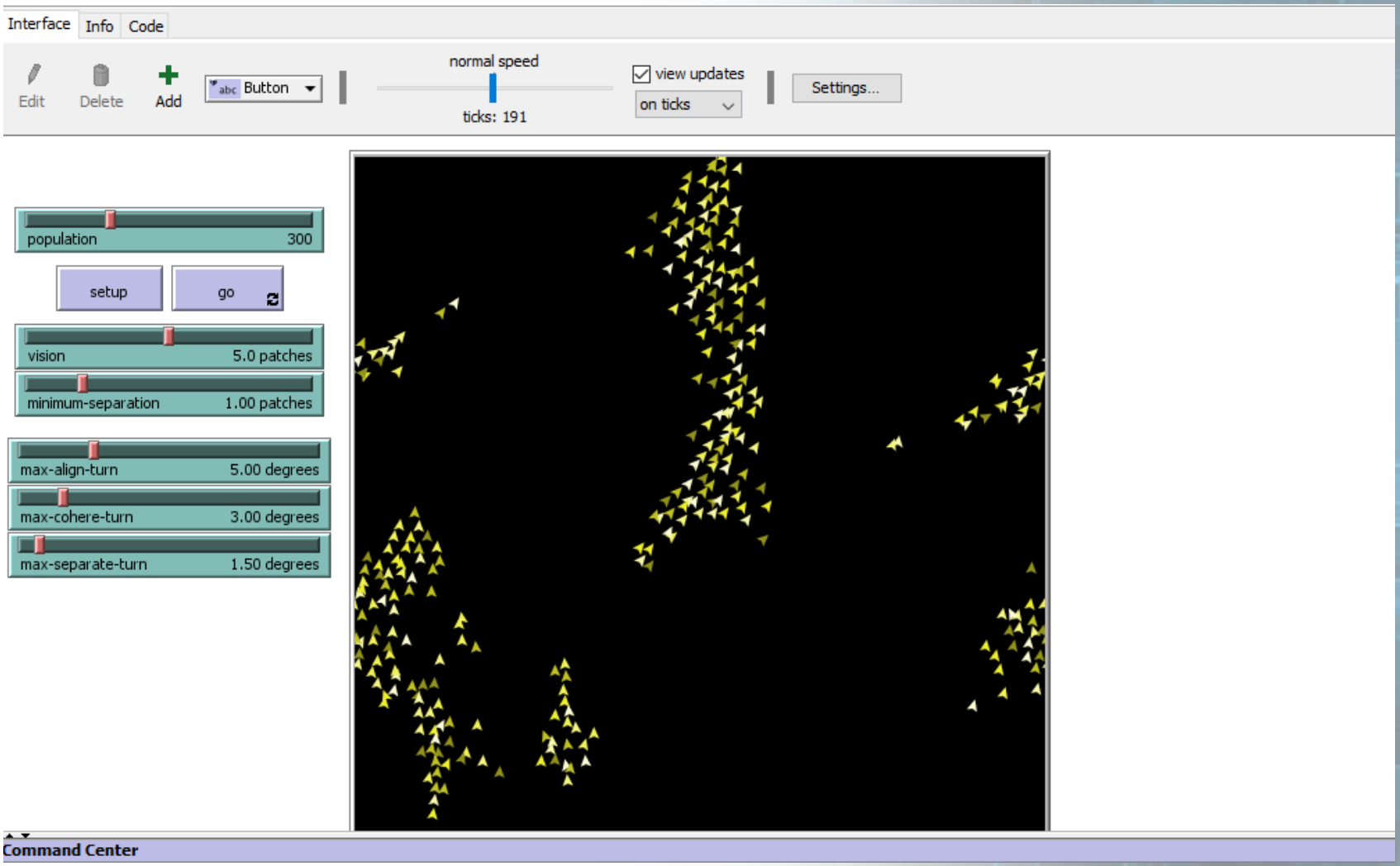
Validation

- **Corresponding between implemented model and reality.**

Validation

- Two axis.
- Macrovalidation.
- Face validation.
- Flocking model.
- A classical agent based model.

Validation


Interface
Info
Code
Edit
Delete
Add
Button
normal speed
ticks: 191
view updates
on ticks
Settings...
population 300
setup
go
vision 5.0 patches
minimum-separation 1.00 patches
max-align-turn 5.00 degrees
max-cohere-turn 3.00 degrees
max-separate-turn 1.50 degrees
Command Center

Validation

```
turtles-own [
  flockmates         ;; agentset of nearby turtles
  nearest-neighbor   ;; closest one of our flockmates
]

to setup
  clear-all
  create-turtles population
    [ set color yellow - 2 + random 7  ;; random shades look nice
      set size 1.5  ;; easier to see
      setxy random-xcor random-ycor
      set flockmates no-turtles ]
  reset-ticks
end

to go
  ask turtles [ flock ]
  ;; the following line is used to make the turtles
  ;; animate more smoothly.
  repeat 5 [ ask turtles [ fd 0.2 ] display ]
  ;; for greater efficiency, at the expense of smooth
  ;; animation, substitute the following line instead:
  ;;   ask turtles [ fd 1 ]
  tick
end
```

Validation

```
to flock  ;; turtle procedure
  find-flockmates
  if any? flockmates
    [ find-nearest-neighbor
      ifelse distance nearest-neighbor < minimum-separation
        [ separate ]
        [ align
          cohere ] ]
end

to find-flockmates  ;; turtle procedure
  set flockmates other turtles in-radius vision
end

to find-nearest-neighbor ;; turtle procedure
  set nearest-neighbor min-one-of flockmates [distance myself]
end

;;; SEPARATE

to separate  ;; turtle procedure
  turn-away ([heading] of nearest-neighbor) max-separate-turn
end

;;; ALIGN

to align  ;; turtle procedure
  turn-towards average-flockmate-heading max-align-turn
end
```

Validation

```
;;; ALIGN

to align  ;; turtle procedure
  turn-towards average-flockmate-heading max-align-turn
end

to-report average-flockmate-heading  ;; turtle procedure
  ;; We can't just average the heading variables here.
  ;; For example, the average of 1 and 359 should be 0,
  ;; not 180.  So we have to use trigonometry.
  let x-component sum [dx] of flockmates
  let y-component sum [dy] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end

;;; COHERE

to cohere  ;; turtle procedure
  turn-towards average-heading-towards-flockmates max-cohere-turn
end

to-report average-heading-towards-flockmates  ;; turtle procedure
  ;; "towards myself" gives us the heading from the other turtle
  ;; to me, but we want the heading from me to the other turtle,
  ;; so we add 180
  let x-component mean [sin (towards myself + 180)] of flockmates
  let y-component mean [cos (towards myself + 180)] of flockmates
  ifelse x-component = 0 and y-component = 0
    [ report heading ]
    [ report atan x-component y-component ]
end
```

Validation

```
;;; HELPER PROCEDURES

to turn-towards [new-heading max-turn]  ;; turtle procedure
  turn-at-most (subtract-headings new-heading heading) max-turn
end

to turn-away [new-heading max-turn]  ;; turtle procedure
  turn-at-most (subtract-headings heading new-heading) max-turn
end

;; turn right by "turn" degrees (or left if "turn" is negative),
;; but never turn more than "max-turn" degrees
to turn-at-most [turn max-turn]  ;; turtle procedure
  ifelse abs turn > max-turn
    [ ifelse turn > 0
        [ rt max-turn ]
        [ lt max-turn ] ]
    [ rt turn ]
end


; Copyright 1998 Uri Wilensky.
; See Info tab for full copyright and license.
```

Validation

summary

- We have understand the validation of agent based model.
- Credits: Uri Wilensky book.

Modeling and Simulation

Macrovalidation vs. Microvalidation

Macrovalidation vs. Microvalidation

Overview

- **Here, we will understand difference between "macrovalidation and microvalidation" and their role in validation of a model.**

Macrovalidation vs. Microvalidation

- ABM's are built from agents hence we can directly compare them with those that exist in the real world.
- For instance, we can ask whether the agents have properties similar to real birds in flocking model.

Macrovalidation vs. Microvalidation

- There are some limitation e.g. real birds can fly in three dimensions , but our bird agents move in two dimension only.
- However, these limitations doesn't make our model invalid.

Macrovalidation vs. Microvalidation

- **we can also build the flocking model in three dimensions and examine the resultant flocks to see if they are relevantly different from the 2D flocks.**

Macrovalidation vs. Microvalidation

- **The other major avenue of validation is to investigate the relationship between the global properties of the model and the flocking patterns of real birds, a process called macrovalidation.**

Macrovalidation vs. Microvalidation

Macrovalidation

- By showing that our model corresponds to the macro-level phenomenon, we further validate that our model is descriptive of real world systems.

Macrovalidation vs. Microvalidation

Conclusion

- **Macrovalidation tells you if you have captured the important parts of the system, whereas microvalidation tells you if you ’ve captured the important parts of the agent ’s individual behavior.**

Macrovalidation vs. Microvalidation

Figure.

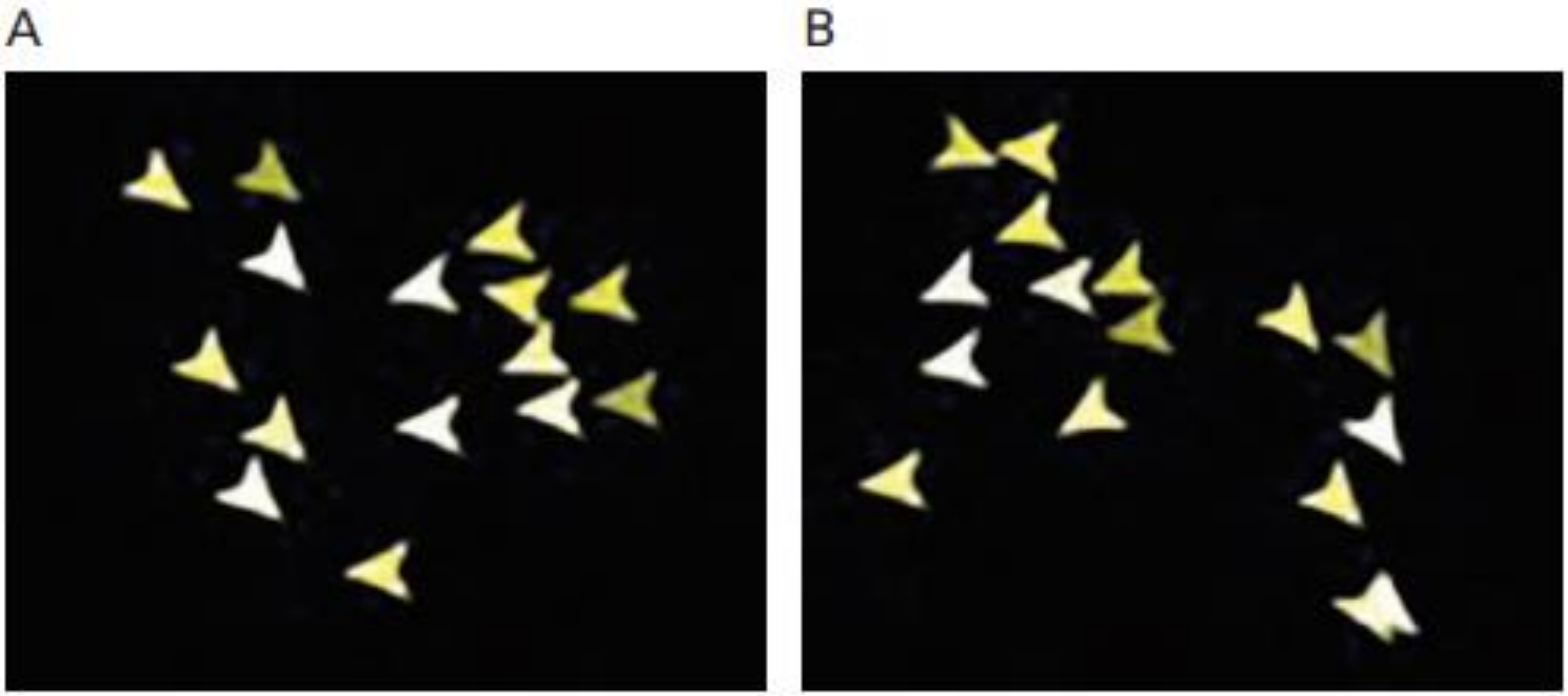


Figure 7.9
Two mini-flocks (A) before and (B) after a collision.

Macrovalidation vs. Microvalidation

Summary

- We understand how macrovalidation and microvalidation compares to each other.
- Credits: Uri Wilensky book.

Modeling and Simulation

Face Validation

vs.

Empirical Validation

Face Validation vs Empirical Validation

Overview

- **Here we will discuss how “face validation” is different from “empirical validation”.**

Face Validation vs Empirical Validation

Face Validation

- **Face validation is the process of showing that the mechanisms and properties of the model look like mechanisms and properties of the real world.**

Face Validation vs Empirical Validation

Empirical Validation

- **Empirical validation is making sure that the model generates data.**
- **Correspond to similar patterns of data in the real world.**

Face Validation vs Empirical Validation

Face Validation

- Face validity can exist at both micro and macro levels.
- Determining whether the birds in the model correspond to real birds is face microvalidity.
- While determining if the flocks correspond to the appearance of real flocks is face macrovalidity.

Face Validation vs Empirical Validation

Empirical Validation

- Empirical validation sets a higher bar.
- Data produced by the model must correspond to empirical data derived from the real-world.
- measures and numerical data generated both by the model and the actual phenomenon.

Face Validation vs Empirical Validation

Empirical Validation

- Inputs and outputs in “ the real world ” are often poorly defined.
- It is hard to isolate and measure parameters from a real-world phenomenon.

Face Validation vs Empirical Validation

Empirical Validation

- **The process of finding the parameters and initial conditions that cause the model to match with real-world data is called calibration.**

Face Validation vs Empirical Validation

Summary

- We understand together, these four types of validation (micro-face, macro-face, micro-empirical, and macro-empirical) characterize the majority of validation efforts carried out.
- Credits: Uri Wilensky book

Modeling and Simulation

Validation Benefits and Questions

Validation Benefits and Questions

Overview

- **Here, we will understand Why is it important to validate a model.**

Validation Benefits and Questions

- A valid model can be useful for extracting general principles about the world.
- Changing the mechanisms and parameters can often help predict what might occur in the real world.

Validation Benefits and Questions

- **A model is not either valid or invalid:**
- **A model can be said to be more or less valid based upon how closely it has been compared to the real process it is modeling.**

Validation Benefits and Questions

- **A model is never inherently valid.**
- **Its validity comes from the context of what it is being used for.**

Validation Benefits and Questions

- **Something in the model corresponds to something in reality.**
- **The user compare his observation's with real world and use it as the basis of his validation.**

Validation Benefits and Questions

Summary

- Here, we understand about the benefits of validation and question it arises.
- Credits: Uri Wilensky book.

Modeling and Simulation

Replication

Replication

Overview

- **Here, we will understand about the “Replication”.**

Replication

- **Model replication is the implementation by one researcher or group of researchers of a conceptual model previously implemented by someone else.**

Replication

- Scientists must publish the details of how the experiment was conducted.
- Then subsequent teams of scientists carry out the experiment themselves to ascertain.

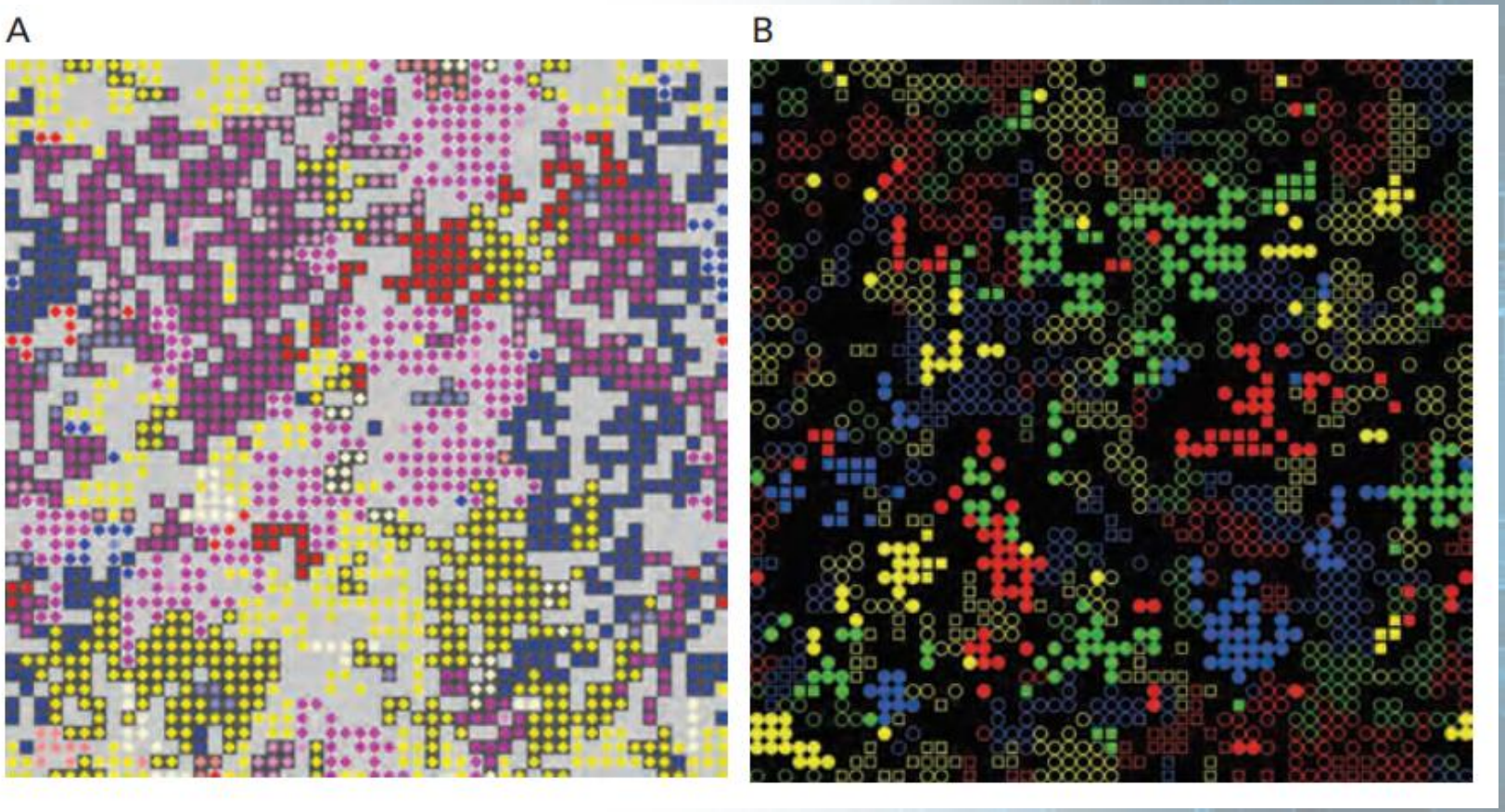
Replication

- Replicating a physical experiment strengthens original results.
- Comparing both the experimental setup and ensuing results.
- Replicating a computational model serves this same purpose.

Replication

- **Replicating a computational model increases our confidence.**
- **New implementation of the model has yielded the same results as the original.**

Replication



Replication

- This model includes a mechanism for inheritance of strategies.
- The model suggests that “ ethnocentric ” behavior can evolve under a wide variety of conditions.
- Even when there are no native “ ethnocentrics ”.

Replication

- Here we understand about the basic concepts of replication and how it is useful for the correctness of a model.
- Credits: Uri Wilensky book

Modeling and Simulation

Replication of Computational Models: Dimensions and Standards

Overview

- **Here, we will understand about the “Replication of Computational Models: Dimensions and Standards”.**

Replication

- **Replication refers to the creation of a new implementation of a conceptual model based on the previous implementation.**

Dimensions

- An original model and replicated model can differ across at least six dimensions.
- Time
- Hardware
- Languages
- Toolkits
- Algorithms
- Authors

Successful Replication

- **A successful replication is one in which the replicators are able to establish that the replicated model creates outputs sufficiently similar to the outputs of the original.**

Replication Standard

- The criterion by which the replication ’s success is judged is called the replication standard.
- Different replication standards exist for the level of similarity between model outputs.

Categories of Replication Standard

- There are three categories of replication standard.
- Numerical identity
- Distributional equivalence.
- Relational alignment.

Data

- ABMs usually produce large amounts of data.
- much of which is usually irrelevant.
- Data that are central to the conceptual model should be measured and tested during replication.

Summary

- Here, we understand about dimensions of conceptual model like time, hardware, toolkit, language, algorithms and author.
- Then we discussed about replication standard and its categories.
- Credits: Uri Wilensky book

Modeling and Simulation

Benefits of replication

Overview

- **We are going to see the benefits of replication in this section.**

Benefits of Replication

- Advances scientific knowledge.
- Model verification.
- Model validation.
- Shared understanding.

Benefits of Replication

- Shared understanding.
- Creation of sets of terms, idioms and best practices.
- Communicate about model.

Benefits of Replication

- Model verification.
- Distinct implementations. producing same results.
- Conceptual model grows by capturing confidence.
- Correction made if there is any difference found in implemented model and replication.

Benefits of Replication

- Model validation.
- Correspondence between the outputs.
- Validation of a model on the basis of output closer to real-world.
- Reevaluating the original mapping.
- Researchers investing and getting interested in it through replication

Benefits of Replication

- Describing the modeling process through developing a language.
- Culture of replication fosters.
- "mean" and "standard deviation" applying them to tests, overtime, replication of ABM experiment to understand "time step" and "shuffled lists".

Summary

- **Here, we understand the benefits of replication in modeling.**
- **Credits: Uri Wilensky book.**

Modeling and Simulation

Recommendations For Model Replicators

Overview

- **Here, we will learn about recommendations for model replicators.**

Recommendations for model replicators

- **RS is to produce the level of precision to establish hypothesis regularity.**
- **Examples : “numerical identity”, “distributional equivalence” and “relational alignment”.**

Recommendations for model replicators

- **How detailed the description of the conceptual model is in the original paper or to communicate with authors or original model.**

Recommendations for model replicators

- **Beneficial to delay the contact with original author until after first attempt to recreate the original model.**
- **Differences between public conceptual model and an implementation can be interesting and resulting in new discoveries.**

Recommendations for model replicators

- Becoming familiar with the toolkit in which original model was written.
- Better understanding of original model.
- Replicator understands the subtler working.
- Deliberately implementing a strategy.

Recommendations for model replicators

- Obtain the source code of original model.
- Effective for illuminating discrepancies in the two model implementations.

Recommendations for model replicators

- **Groupthink**
- **Unconsciously adopting some of the practices of the original model developer.**

Recommendations for model replicators

Table 7.1
Details to be included in published replications. For each category, sample options to choose from are listed.

Categories of Replication Standards:
Numerical Identity, Distributional Equivalence, Relational Alignment
Focal Measures:
Identify particular measures used to meet goal
Level of Communication:
None, Brief email contact, Rich discussion and personal meetings
Familiarity with Language/Toolkit of Original Model:
None, Surface understanding, Have built other models in this language/toolkit
Examination of Source Code:
None, Referred to for particular questions, Studied in-depth
Exposure to Original Implemented Model:
None, Reran original experiments, Ran experiments other than original ones
Exploration of Parameter Space:
Examined results from original paper, Examined other areas of the parameter space

Summary

- We have understand the recommendation for model replicators in modeling.
- Credits: Uri Wilensky book.

Modeling and Simulation

Recommendations for Model Authors

Overview

- **Here, we will understand the recommendations for the model authors.**

Recommendations for the model authors

- Research paper should be well specified.
- Complete source code for the model may not sufficient.
- Model authors must make their source code publicly available or at least the “pseudo code”.

Recommendations for the model authors

- To what extent the model developer presents a sensitivity analysis.
- Small modification to the original model can effect results drastically.
- These sensitive differences should be published by the authors.

Recommendations for the model authors

- Original author may not be the original implementer.
- Implementation will not be veridical.
- Model authors implement their own models using “ low-threshold ” and toolkits.

Recommendations for the model authors

- **Authors to examine their conceptual models through the lens of a potential model replicator.**

Summary

- **We have understand the recommendations for model authors in modelling.**
- **Credits: Uri Wilensky book**

END

Modeling and Simulation

Complex Adaptive Systems

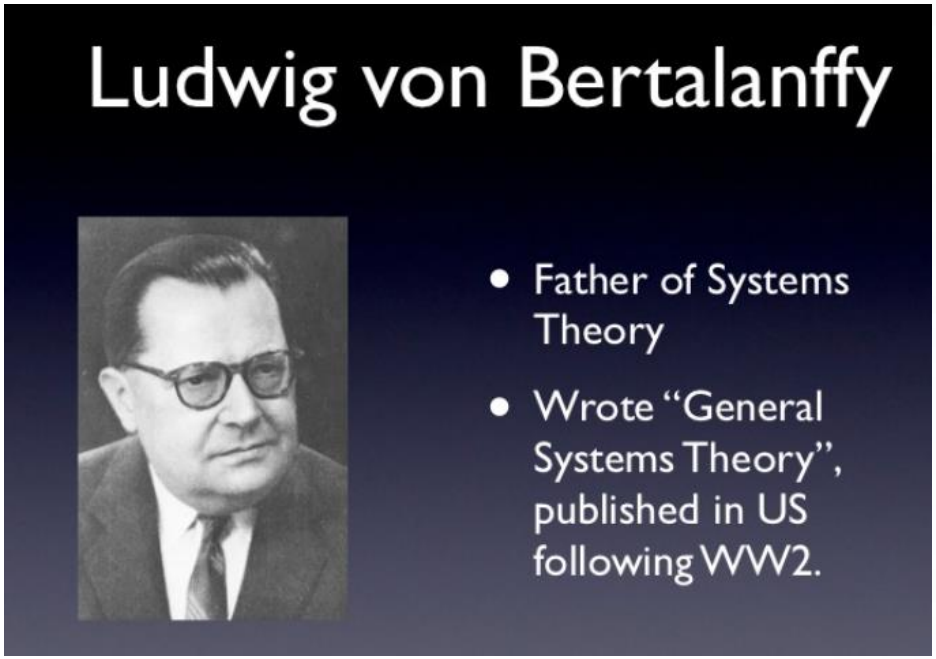
Overview

- **Here, we will understand the basics about complex adaptive systems (CAS).**

Complex Adaptive Systems

- Complexity
- Adaptation
- Systems

Complex Adaptive Systems



- **The systems approach is an old concept.**
- **The approach stands on the assumption that breaking down of a complex concept into simple easy to understand units helps in better understanding of the complexity.**
- **Ludwig von Bertalanffy first proposed it as 'General System Theory'**

Complex Adaptive Systems

- A system is a delineated part of the universe which is distinguished from the rest by an imaginary boundary. (NECSI)
- Properties of the system, the properties of the universe excluding the system which affect the system and the interactions / relationships.

Complex Adaptive Systems

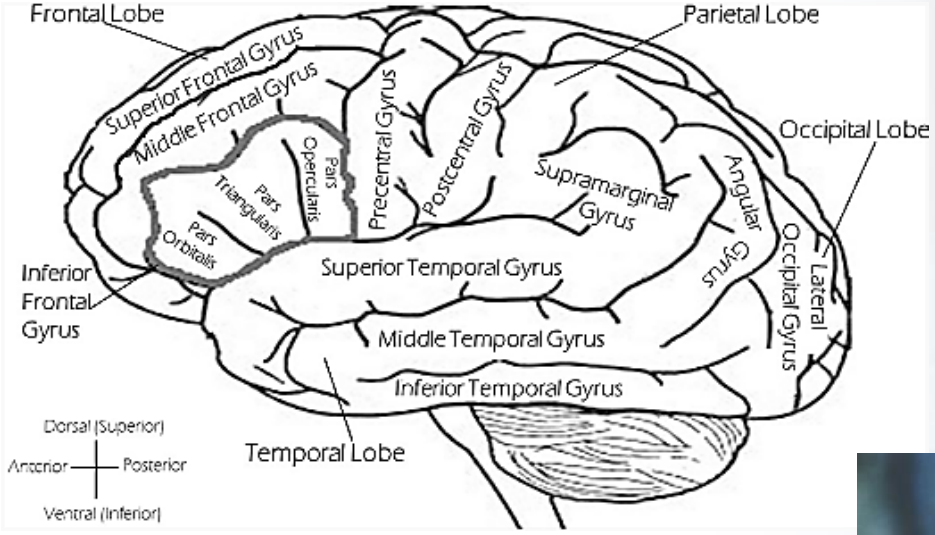
- An adaptive system (or a complex adaptive system, CAS) is a system that changes its behavior in response to its environment.
- The adaptive change that occurs is often relevant to achieving a goal or objective. (NECSI)

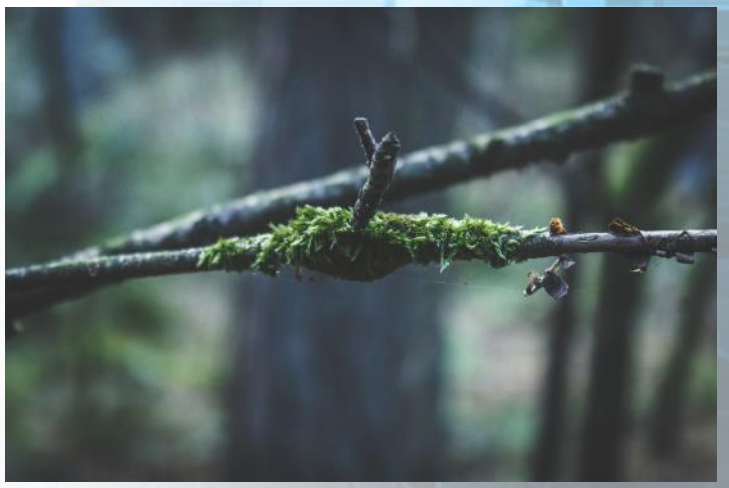
Complex Adaptive Systems

- Complexity is:
- ...the (minimal) length of a description of the system.
- ...the (minimal) amount of time it takes to create the system.
- Here the length of a description is measured in units of information.

Complex Adaptive Systems

Examples





Summary

- We have learnt about the basic ideas of Complex Adaptive Systems.
- Credits: Uri Wilensky Book.

Modeling and Simulation

History

Overview

- **Here, we will understand about the historical perspective of CAS**

History

- **Started with the concepts of General Systems Theory**

History

- **Biologists focused primarily on individual cells**

History

- **Social Scientists focused on connections**

History

- **John Holland and cas/CAS**

Summary

- We have learnt about the historical perspective of CAS.
- Credits: Niazi & Hussain book.

Modeling and Simulation

Complexity

Overview

- **Here, we will understand about the basic ideas of complexity**

Complexity

- **Disambiguation from Computational Complexity**

Complexity

- **Nonlinearity**
- **Large number if variables**

Complexity

- Chaos theory
- The term "chaos" is popularly used to refer to disorder or confusion.
- In science, chaos is an important conceptual paradox that has a precise mathematical meaning:
- A chaotic system is a deterministic system that is difficult to predict.

Complexity

- A deterministic system is defined as one whose state at one time completely determines its state for all future times.
- So what does it mean for a chaotic system to be difficult to predict?
- Butterfly effect

Summary

- We have learnt about the basics of Complexity in CAS.
- Credits: Niazi and Hussain book.

Modeling and Simulation

Adaptation

Overview

- **Here, we will understand about adaptation in CAS.**

Adaptation

- **Adaptive changes in interactions**

Adaptation

- **Who changes**

Adaptation

- **What changes**

Adaptation

- **When changes**

Adaptation

- **How changes**

Summary

- We have learnt about the adaptation in CAS.
- Credits: Niazi and Hussain book.

Modeling and Simulation

Systems Approach

Overview

- **Here, we will understand about the Systems approach.**

Systems Approach

- **Concept of a System - boundaries**

Systems Approach

- Many systems can be correlated
- Example: Life/Humans/Society

Systems Approach

- **Whole vs. parts**

Systems Approach

- **Reductionist vs. Wholistic approach**

Summary

- We have learnt about the systems perspective of things.
- Credits: Niazi and Hussain book.

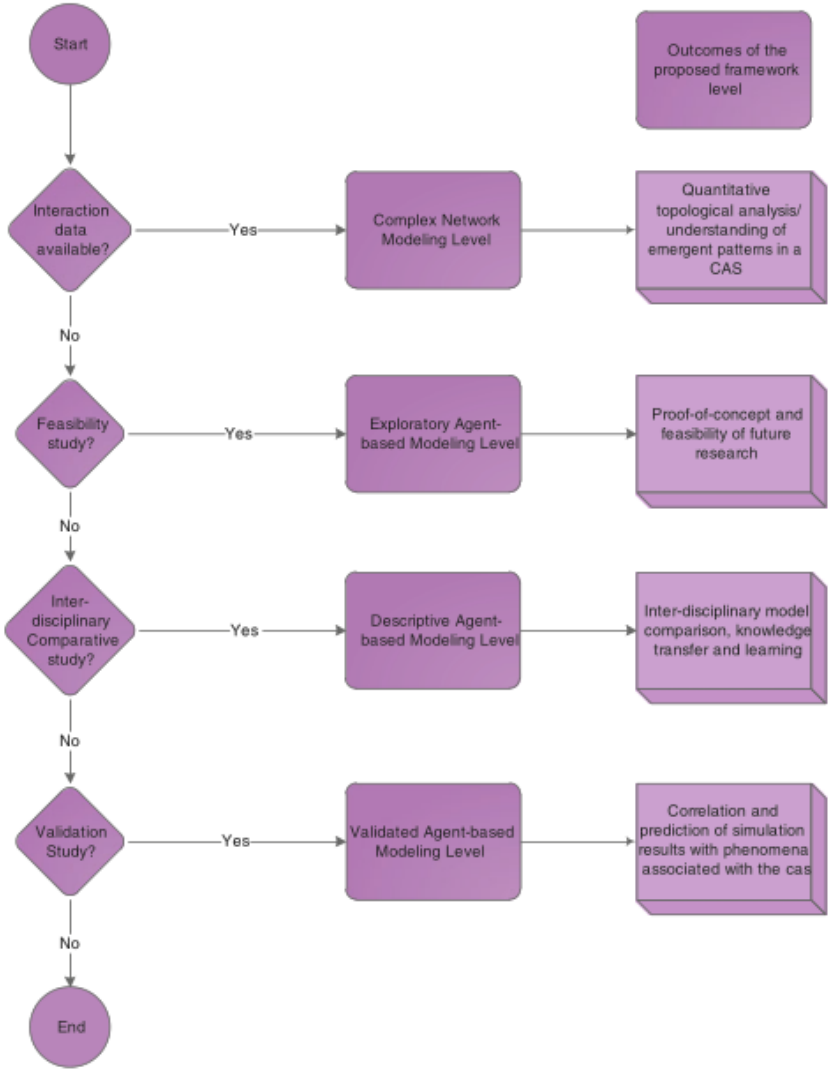
Modeling and Simulation

Modeling of CAS

Overview

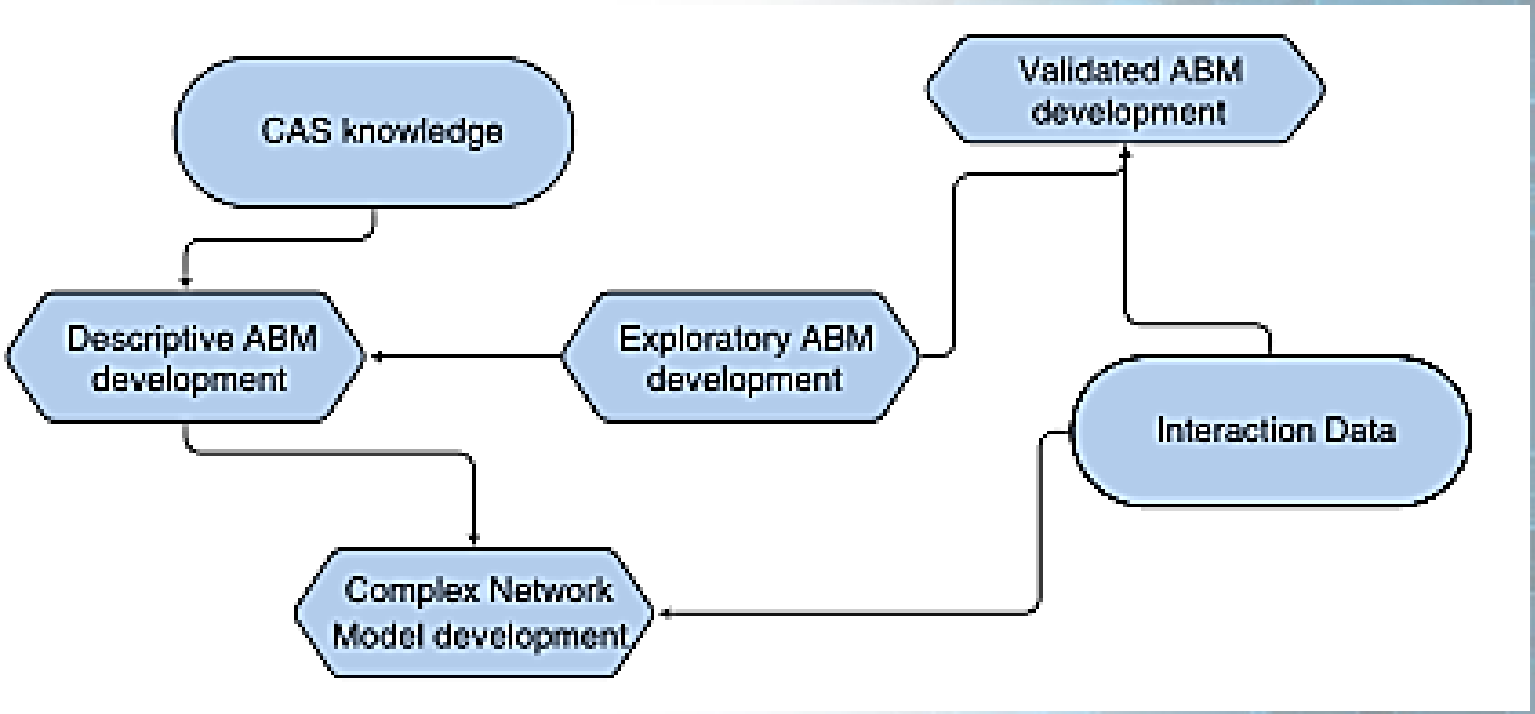
- **Here, we will understand about the problems encountered in modeling of CAS.**

Modeling of CAS



- **Cognitive Agent-based (CABC) Framework**

Modeling of CAS



Summary

- We have learnt about the modeling of CAS using CABC framework
- Credits: Niazi and Hussain book.

Modeling and Simulation

Agent-based Approach

Overview

- **Here, we will understand about the agent-based approach to modeling**

Agent-based Approach

- Agent – something that acts
- There is a concept of agent in domains as distinct as:
- AI/CS/Robotics
- Sociology (Individual)
- Biology
- Ecology
- Etc.

Agent-based Approach

- CABC framework has 3 levels for Agent-based modeling
- Exploratory Agent-based modeling (EABM)
- Descriptive Agent-based Modeling (DREAM)
- Validated Agent-based Modeling (VOMAS-based Modeling or VABM)

Agent-based Approach

- **Exploratory ABM**

Agent-based Approach

- **DREAM**

Agent-based Approach

- Virtual Overlay Multiagent System (VOMAS)
- Validated Agent-based Modeling

Summary

- We have learnt about various approaches for modeling using Agents and the CABC framework.
- Credits: Niazi and Hussain book.

Modeling and Simulation

Complex Network-based Modeling

Overview

- **Here, we will understand about complex networks/graphs and modeling using the CABC Level 1**

Complex Network-based Modeling

- What is a graph?
- Graph/Network
- Same entity different terms in different domains

Complex Network-based Modeling

- **Network = weighted graph**

Complex Network-based Modeling

- Different aspects of modeling using networks
- Labels
- arcs/lines
- Directed/Undirected
- Centrality-based
- Ego
- Sentiment graphs
- Community Detection

Complex Network-based Modeling

- Key benefits
- Very well-defined
- Lot of free tools
- Lot of community support
- Well-defined mathematical models

Complex Network-based Modeling

- Cons
- Difficult to figure out
- Difficulty in graph mining – lack of expertise in the community
- Missing data can be very difficult
- Requires complete data
- Complexity in deciding data boundaries

Summary

- We have learnt about complex-network modeling using the CABC framework
- Credits: Niazi and Hussain book.

Modeling and Simulation

Cognitive Agent-based Computing Framework

Overview

- **Here, we will understand how to put modeling and simulation to use employing the CABC framework.**

Cognitive Agent-based Computing Framework

- Data and Models
- How much data is needed

Cognitive Agent-based Computing Framework

- **Why choose Agent-based Modeling?**

Cognitive Agent-based Computing Framework

- **Why choose Complex network modeling?**

Cognitive Agent-based Computing Framework

- **When to choose EABM?**

Cognitive Agent-based Computing Framework

- **When to choose DREAM?**

Cognitive Agent-based Computing Framework

- **When to choose VOMAS?**

Summary

- We have learnt about putting the CABC framework to work.
- Credits: Niazi and Hussain book.

Modeling and Simulation

Useful Statistical Models

Overview

- **Here, we will learn about the useful statistical models.**

Useful Statistical Model

- The models that are useful in the case of limited data are:
- Queueing systems.
- Inventory and supply-chain systems.
- Reliability and maintainability.
- Limited data.
- Other distributions.

Queueing Systems

- Simulation solved the waiting line problems.
- Interval and service time patters were given in queuing examples.
- In these examples service and arrival time was probabilistic.
- But it is possible to have constant arrival time and constant service time.

Queueing Systems

- “Arrivals” can occur due to many reasons like machine break downs.
- Exponential distribution used to simulate the random exponential distribution of services.
- For special cases truncated normal distribution can be utilized.

Inventory and supply-chain systems

- Three random variables
- Number of units demanded per order or per time period.
- Time between demands.
- Lead time.
- Simple inventory systems have constant demand and lead time constant or zero.

Inventory and supply-chain systems

- **Mathematical tractability based demand and lead time in inventory theory text could be invalid.**
- **Variety of demand patterns are satisfied by geometric, Poisson and negative binomial distributions.**

Reliability and Maintainability

- Failure time is modeled with many distributions.
- When failure occurs randomly model distribution is considered as exponential.
- Modeling standby redundancy required for gamma distribution.

Limited Data

- Three distributions.
- Uniform distribution
- Triangular distribution
- Beta distribution
- Uniform distribution is just a special case of beta distribution

Summary

- Here, we discussed about useful statistical model, queuing systems, Inventory and supply-chain systems, Limited data, Reliability, and maintainability.
- Credits: Jerry Banks book.

Modeling and Simulation

Bernoulli Distribution

Overview

- **Here, we will learn about the Bernoulli distribution.**

Bernoulli Distribution

- **Bernoulli trail is one of the simplest experiment you can do in statistic and probability which has one out of two possible outcome.**
- **Bernoulli distribution is a discrete probability distribution for Bernoulli trail which has one out of two possible outcomes success or fail.**

Bernoulli Distribution

- Let us consider an example which consists of n trails and each of which can be a success or a failure.
- $X_j = 1$ if the jth experiment is success
- $X_j = 0$ if its failure.
- N trails and each trail has only two out comes success or failure.

Bernoulli Distribution

$$p(X_1, X_2, \ldots\ldots, X_n) = p_1(X_1) \cdot p_2(X_2) \cdot \cdot \cdot p_n(X_n)$$

$$p_j(x_j) = p(x_j) = \begin{cases} p, & x_j = 1, j = 1, 2, \ldots, n \\ 1 - p = q, & x_j = 0, j = 1, 2, \ldots, n \\ 0, & \text{otherwise} \end{cases}$$

- **Success here happens with probability (p) and failure at (p – 1).**
- **X is a discrete random variable. It takes value 1 in case of success and 0 in case of failure.**

Bernoulli Distribution

- **Below is the expected value of random variable X is calculated as:**

$$E(X_j) = 0 \cdot q + 1 \cdot p = p$$

- **Below is the variance of the random variable X is calculated as:**

$$V(X_j) = \left[\left(0^2 \cdot q\right) + \left(1^2 \cdot p\right)\right] - p^2 = p(1-p)$$

Bernoulli Distribution

- We have understand about the Bernoulli distribution.
- Credits: Jerry Banks book.

Modeling and Simulation

Binomial Distribution

Overview

- **Here, we will learn about the binomial distribution.**

Binomial Distribution

- **Binomial distribution is the discrete probability distribution with parameters n and p of the number of successes in the sequence of n independent experiments.**
- **Each n and p with its own boolean-value.**
- **Random variable.**

Binomial Distribution

- **Bernoulli trial's random variable X has binomial distribution:**

$$p(x)=\begin{cases}\binom{n}{x}p^{x}q^{n-x}, & x=0,1,2,\ldots,n\\ 0, & \text{otherwise}\end{cases}$$

- **This equation is for calculating the probability of a specific outcome with all successes.**

Binomial Distribution

$$P(\overbrace{SSS\ldots\ldots SS}^{x \text{ of these}}\overbrace{FF\ldots\ldots FF}^{n-x \text{ of these}}) = p^x q^{n-x}$$

- S = success.
- In first X trials.
- $n - x = \mathrm{f}$ (failures).

Binomial Distribution

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$

- $q = 1 - p$
- **This equation shows the required number of successes and failures.**

Binomial Distribution

$$p(x)=\begin{cases}\binom{n}{x}p^x q^{n-x}, & x=0,1,2,\ldots,n \\ 0, & \text{otherwise}\end{cases}$$

- Again this equation can be used to calculate mean and variance of binomial distribution.
- X = sum of independent Bernoulli random variables. Then:

$$X = X_1 + X_2 + \cdots + X_n$$

Bernoulli Distribution

- Expected value $E(X)$

$$E(X) = p + p + \cdots + p = np$$

- n = total number of experiments.
- p = probability of each experiment getting successful result.
- X = binomially random variable.

Bernoulli Distribution

- Variance $V(X)$

$$V(X) = pq + pq + \cdots + pq = npq$$

- n = total number of experiments.
- p = probability of each experiment getting successful result.
- X = binomially random variable.
- q = probability of failure in each experiment.

Summary

- We have understand about the Binomial distribution.
- Credits: Jerry Banks book.

Modeling and Simulation

Poisson Distribution

Overview

- **Here, we will learn about the Poisson distribution.**

Poisson Distribution

- **A discrete frequency distribution that gives the probability of independent events occurring in a fixed interval of time is known as the Poisson distribution.**

Poisson Distribution

$$p(x) = \begin{cases} \dfrac{e^{-\alpha}\alpha^{x}}{x!}, & x = 0, 1, \ldots \\ 0, & \text{otherwise} \end{cases}$$

- X = 0,1,2,3,….
- $e = 2.71828$
- $a =$ mean number of successes in the given time interval or region of space.
- $a > 0$

Poisson Distribution

$$E(X) = \alpha = V(X)$$

- **This is the important property of Poisson distribution that is mean and variance is equal to α.**

Poisson Distribution

$$F(x) = \sum_{i=0}^{x} \frac{e^{-\alpha}\alpha^{i}}{i!}$$

- **This is the cumulative distribution function.**

Summary

- We have understand the Poisson distribution.
- Credits: Jerry Banks book.

Modeling and Simulation

Uniform Distribution

Overview

- **Here, We will understand “Uniform Distribution” also known as “rectangular distribution”.**

Uniform Distribution

- **The uniform distribution is a continuous distribution.**
- **It takes values within a specified range.**

Uniform Distribution

- **Probability density function for a uniform distribution always take a random number on [0,1].**

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Uniform Distribution

- **Consider a random variable X on the interval [a, b] distributed uniformly. The, equation is given by:**

- $X = a + (b - a)R$

Uniform Distribution

- $X = a + (b - a)R$
- For the derivative of function.
- First, compute the cdf of variable X.
- Second, set $F(x) = R$ on the range of X.
- Third, solve the equation $\mathrm{F}(x)$ in terms of R for X.

Uniform Distribution

- **Step 1:**
- **The cdf for the equation is :**

$$F(x) = \begin{cases} 0 & : x < 0 \\ x & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$

- **If X takes only discrete values 0 and 1 then:**

$$F(x) = \begin{cases} 0 & : x < 0 \\ 1/2 & : 0 \leq x < 1 \\ 1 & : x \geq 1. \end{cases}$$

Uniform Distribution

- **Step 2:**
- **Set**

$$F(x) = (X-a)(X-b)R$$

- **Step 3:**
- **Solve for X in terms of R which proves.**

$$X = a + (b-a)R$$

Summary

- Here, we understood about the fundamentals of Uniform Distribution.
- Credits, Discrete Event System Simulation, Jerry Banks.

Modeling and Simulation

Exponential Distribution

Overview

- **Here, we will understand About “Exponential Distribution”.**

Exponential Distribution

- **The Exponential Distribution is a continuous valued probability distribution.**
- **Takes positive real values and λ which controls the shape of the distribution.**

Exponential Distribution

- **The probability density function**

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- **This can also be defined as right-continuous Heaviside step function.**

$$f(x;\lambda) = \lambda e^{-\lambda x} H(x)$$

- **Here λ is called the rate parameter.**

Exponential Distribution

- The cumulative distribution function

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- This can also be defined as heaviside step function

$$F(x;\lambda) = (1 - e^{-\lambda x})H(x)$$

- Here λ is the number of occurrences per time unit.

Exponential Distribution

- **Step 1:**
- **Compute the cdf of random variable x given**

$$F(x) = 1 - e^{-\lambda x}, \; x \geq 0.$$

- **Step 2**
- **Set $F(x) = R$ on the range of x which becomes as.**

$$1 - e^{-\lambda x} = R \text{ on the range } x \geq 0.$$

Exponential Distribution

- **Step 3**
- **Solve the equation $F(X) = R$ for X in terms of R as follows.**

$$1 - e^{-\lambda X} = R$$

$$e^{-\lambda X} = 1 - R$$

$$-\lambda X = \ln(1 - R)$$

$$X = -\frac{1}{\lambda}\ln(1 - R)$$

- **Here, X is called random-variate generator.**

Exponential Distribution

- **Step 4**
- **Generate uniform random numbers to compute desired random variates.**

$$X_i = F^{-1}(R_i)$$

- **For exponential case.**

$$X_i = -\frac{1}{\lambda}\ln(1-R_i)$$

Exponential Distribution

- If $i = 1, 2, 3 \ldots$ replace $1 - Ri$ with R_i as follows.

$$X_i = -\frac{1}{\lambda} \ln R_i$$

- Which proves that both R_i and $1 - R_i$ are uniformly distributed on [0, 1].

Summary

- We understand about “Exponential Distribution”
- Credits, Discrete Event System Simulation, Jerry Banks.

Modeling and Simulation

Triangular Distribution

Overview

- **Here, we will discuss about “Triangular Distribution”.**

Triangular Distribution

- A triangular distribution is a continuous probability distribution with a probability density function shaped like a triangle defined by
- The minimum value a
- The maximum value b
- The peak value c
- Where $a < b$ and $a \leq c \leq b$

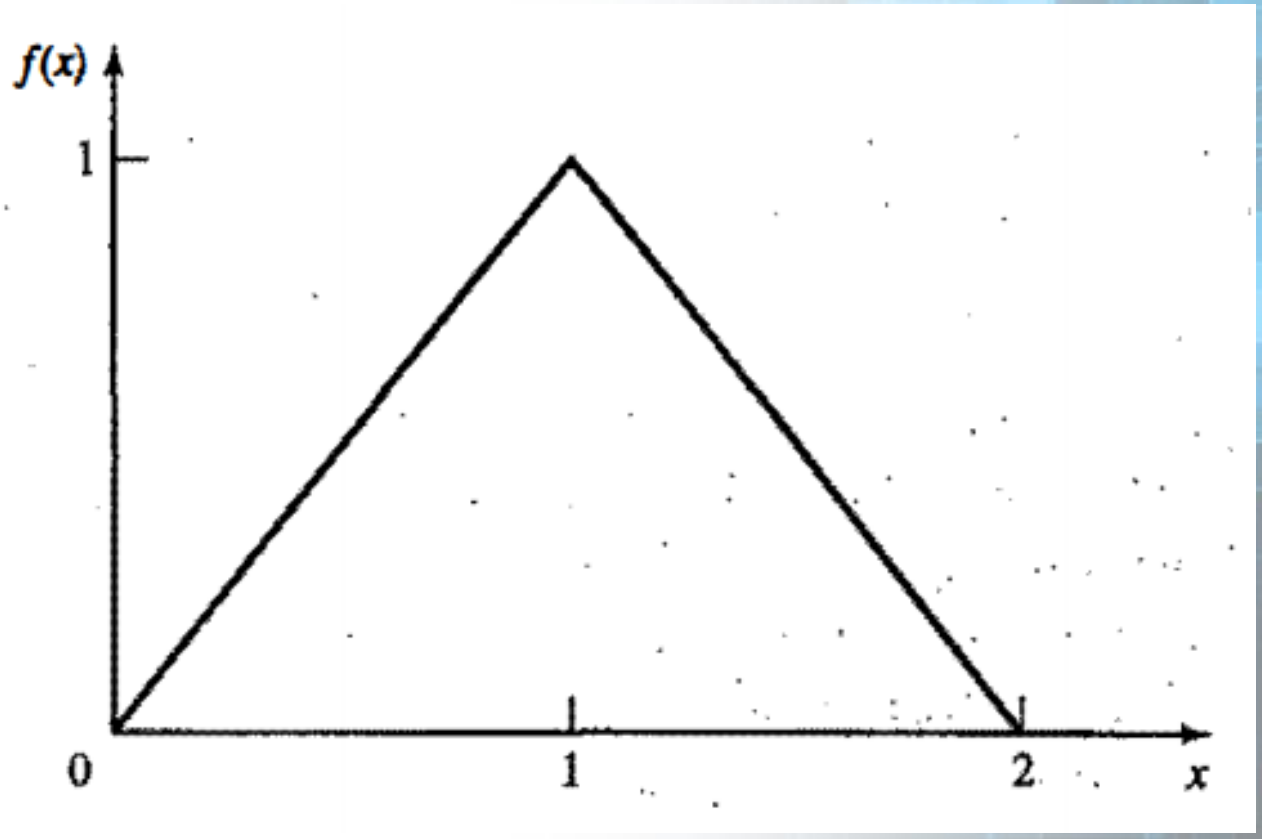
Triangular Distribution

- **Probability density function with variable X**

$$f(x)=\begin{cases} x, & 0\le x\le 1 \\ 2-x, & 1<x\le 2 \\ 0, & \text{otherwise} \end{cases}$$

- **Where endpoints are $(0, 2)$ and mode at 1.**

Triangular Distribution



Triangular Distribution

- **Its cumulative distribution function is given as :**

$$F(x)=\begin{cases}0, & x\le 0\\ \dfrac{x^2}{2}, & 0<x\le 1\\ 1-\dfrac{(2-x)^2}{2}, & 1<x\le 2\\ 1, & x>2\end{cases}$$

Triangular Distribution

- **For** $0 \leq X \leq 1$:

$$R = \frac{x^2}{2}$$

- **Which implies that**

$$x = \sqrt{2R}$$

Triangular Distribution

- **For** $1 \leq X \leq 2$:

$$R = 1 - \frac{(2 - X^2)}{2}$$

- **Which implies that**

$$x = 2 - \sqrt{2(1 - R)}$$

Triangular Distribution

- Thus X is generated as:

$$X = \begin{cases} \sqrt{2R}, & 0 \le R \le \frac{1}{2} \\ 2 - \sqrt{2(1-R)}, & \frac{1}{2} < R \le 1 \end{cases}$$

Summary

- **Here, we have understood the basics of Triangular Distribution, its pdf, and cdf.**

CHARACTERISTICS OF QUEUEING SYSTEMS

Overview

- **Here, we will learn about characteristics of queueing systems.**

CHARACTERISTICS OF QUEUEING SYSTEMS

- Customers and servers are the key elements of a queueing system.
- A Customer is anything that visits a facility and requires some service.
- Servers the service provider.

CHARACTERISTICS OF QUEUEING SYSTEMS

- **Not every time the customer visits the service facility.**

CHARACTERISTICS OF QUEUEING SYSTEMS

Table 6.1 Examples of Queueing Systems

System	*Customers*	*Server(s)*
Reception desk	People	Receptionist
Repair facility	Machines	Repairperson
Garage	Trucks	Mechanic
Tool crib	Mechanics	Tool-crib clerk
Hospital	Patients	Nurses
Warehouse	Pallets	Fork-lift Truck
Airport	Airplanes	Runway
Production line	Cases	Case-packer
Warehouse	Orders	Order-picker
Road network	Cars	Traffic light
Grocery	Shoppers	Checkout station
Laundry	Dirty linen	Washing machines/dryers
Job shop	Jobs	Machines/workers
Lumberyard	Trucks	Overhead crane
Sawmill	Logs	Saws
Computer	Jobs	CPU, disk, CDs
Telephone	Calls	Exchange
Ticket office	Football fans	Clerk
Mass transit	Riders	Buses, trains

Summary

- We have learnt about the characteristics of the Queueing systems.
- Credits: Jerry Banks books.

Modeling and Simulation

The Calling Population

Overview

- **Here, we will learn about the calling population.**

The Calling Population

- Calling population is the population of potential customers.
- Finite or infinite.
- Example of the bank having five machines and each machine is a customer and worker is its server.

The Calling Population

- **Infinite calling population when the system has a large population of potential.**
- **Examples are banks and restaurants.**

The Calling Population

- Arrival rate is the main difference between the finite and infinite population models.
- Infinite population system is not affected by the arrival rate.
- The Finite system depends on the no of customers waiting and being served.

Summary

- We have learnt about the calling population and what are effects of finite and infinite population.
- Credits: Jerry Banks book.

Modeling and Simulation

System Capacity

Overview

- **Here, we will learn about the system capacity.**

System Capacity

- Every queueing system has some capacity.
- They have different number of customers entering the system.

System Capacity

- There is a limit for every queueing system to hold the number of customers.
- If the entering system finds the capacity full it will return to the calling population.

System Capacity

- Some systems have unlimited capacity
- Like tickets sold for a concert to students and many students can wait to purchase them.

System Capacity

- **In limited systems, there is a distinction in arrival time and effective arrival time.**

Summary

- We have learnt about the system capacity.
- Credits: Jerry Banks book.

Modeling and Simulation

The Arrival Process

Overview

- **Here, we will learn about the arrival process.**

Arrival Process

- For infinite population models the arrival process characterized in terms of interarrival times of successive customers.
- Probability distribution is used to characterize the interarrival times when arrival is occurred in random time.

Arrival Process

- Poisson arrival process is the most important for random arrivals.
- If A_n = inter-arrival
- Between n-1(customer) and
- n(customer) then
- A_n is exponentially distributed with mean $1/\lambda$ time units.

Arrival Process

- The arrival rate is λ customers per time unit.
- So the number of arrivals in a time interval of length t has Poisson distribution with mean λt customers.
- In restaurants and banks the Poisson arrival process has been employed successfully.

Arrival Process

- Scheduled arrival is the second arrival class.
- Inter-arrival could be either constant or constant plus or minus a small random amount to represent early or late arrivals.

Arrival Process

- Third situation occurs when one customer is always present in the queue.
- So we do not have an idle server.

Arrival Process

- For finite population the arrival process is different.
- We define the customer as pending when he is out of the queue system and a member of potential calling population.

Arrival Process

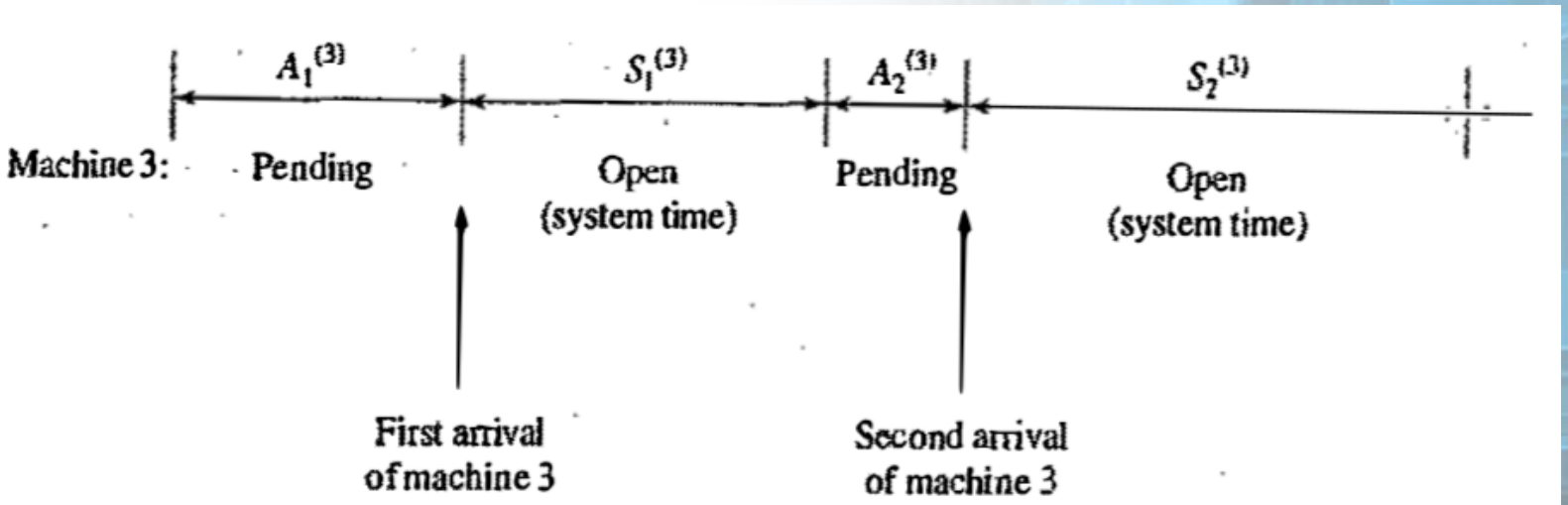


Figure 6.2 Arrival process for a finite-population model.

Arrival Process

- **Important application of finite population model is machine repair problem.**

Summary

- We have learnt about the finite and infinite arrival processes.
- Credits: Jerry Banks book.

Modeling and Simulation

QUEUE BEHAVIOR AND QUEUE DISCIPLINE

Overview

- **Here, we will learn about the Queue behavior and Queue discipline.**

Queue Behavior

- The actions of the customer while waiting in the queue system for his service is known as the queue behaviour.
- The customers can baulk, renege or jockey.

Queue Discipline

- The logical ordering of the customers in a queue and determining that which customer should come for service when the server is free.
- FIFO, LIFO, SIRO, SPT and PR are some common queue disciplines.

Summary

- We have learnt about the Queue behavior and Queue discipline.
- Credits: Jerry Banks book.

Modeling and Simulation

Service Times and the Service Mechanism

Overview

- **Here, we will learn about the Service Times and the Service Mechanism.**

Service Times and the Service Mechanism

- Service times of successive arrivals are denoted by S_1 S_2…..
- Their duration can be random or constant.
- This latter case is characterized as a sequence of random or identically distributed variables.

Service Times and the Service Mechanism

- The exponential, Weibull, gamma, lognormal and truncated normal distributions are used for service times.
- Sometimes the services are identically distributed for all same customers. Same type customers can have different service time.

Service Times and the Service Mechanism

- Number of service centers and interconnecting queues are present in a queue system.
- Parallel working servers for each service centre.

Service Times and the Service Mechanism

- Server = c
- Single server (c = 1)
- Multiple servers ($1<c<\infty$)
- Unlimited servers($c = \infty$)

Service Times and the Service Mechanism

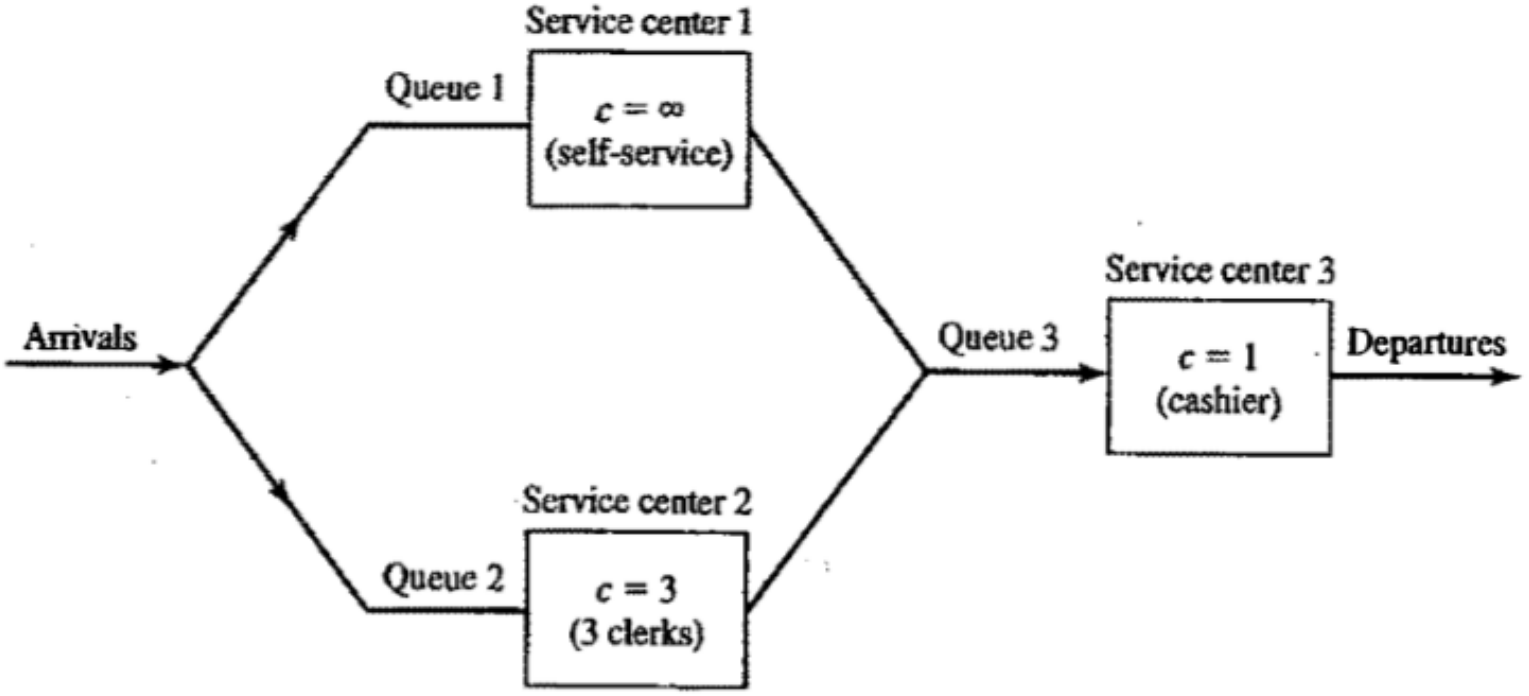


Figure 6.3 Discount warehouse with three service centers.

Service Times and the Service Mechanism

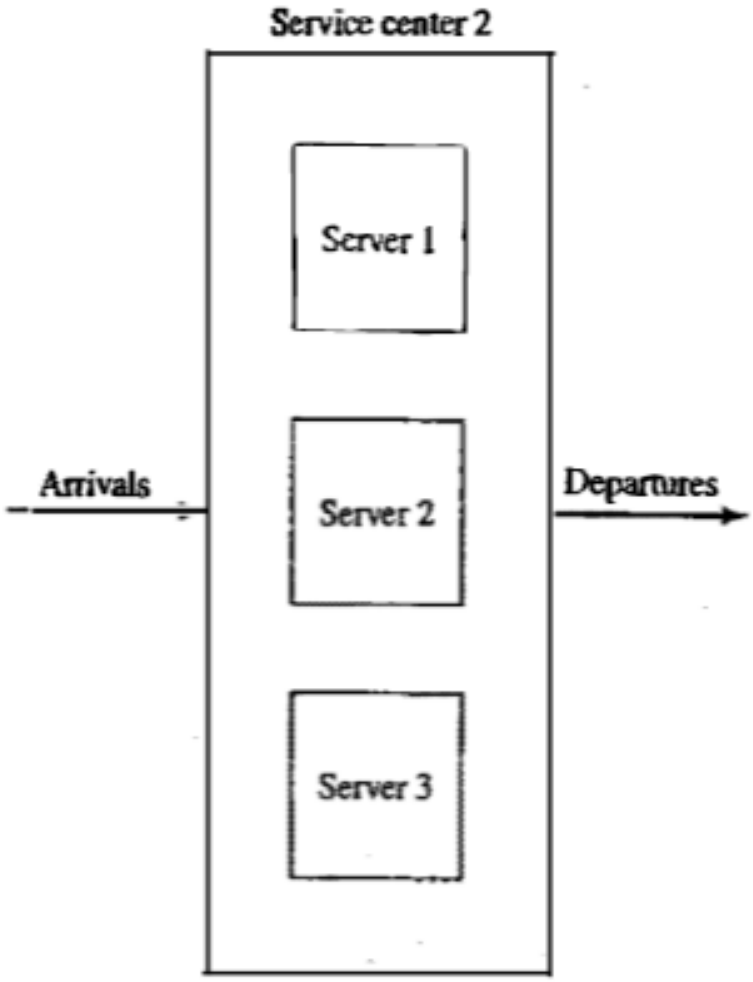


Figure 6.4 Service center 2, with $c = 3$ parallel servers.

Service Times and the Service Mechanism

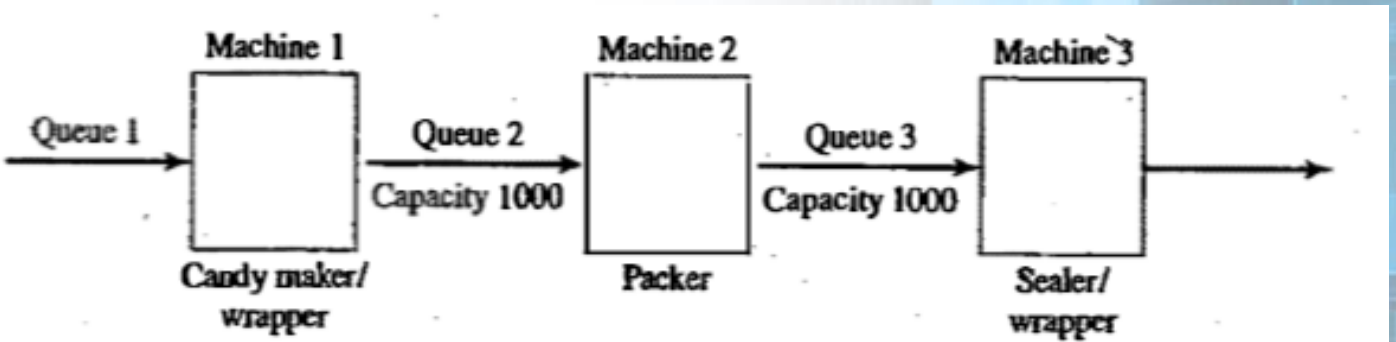


Figure 6.5 Candy-production line.

Summary

- We have learnt about the Service times and the Service mechanism.
- Credits: Jerry Banks book.

Modeling and Simulation

Queueing Notation

Overview

- **Here, we will learn about the Queueing notation.**

Queueing Notation

- Proposed by Kendall [1953]
- For parallel server systems.
- A/B/c/N/K based on this format.
- “A” represents the inter-arrival-time distribution.
- “B” represents the service-time distribution.

Queueing Notation

- **“C” represents the number of parallel servers.**
- **“N” represents the system capacity.**
- **“K” represents the size of the calling population.**

Queueing Notation

- Common symbols for A and B include M (exponential or Markov)
- D (constant or deterministic).
- Ek (Erlang of order k).

Queueing Notation

- PH (phase-type).
- H (hyperexponential).
- G (arbitrary or general)
- Gl (general independent).

Queueing Notation

Table 6.2 Queueing Notation for Parallel Server Systems

P_n	Steady-state probability of having n customers in system
$P_n(t)$	Probability of n customers in system at time t
λ	Arrival rate
λ_e	Effective arrival rate
μ	Service rate of one server
ρ	Server utilization
A_n	Interarrival time between customers $n-1$ and n
S_n	Service time of the nth arriving customer
W_n	Total time spent in system by the nth arriving customer
W_n^Q	Total time spent in the waiting line by customer n
$L(t)$	The number of customers in system at time t
$L_Q(t)$	The number of customers in queue at time t
L	Long-run time-average number of customers in system
L_Q	Long-run time-average number of customers in queue
w	Long-run average time spent in system per customer
w_Q	Long-run average time spent in queue per customer

Summary

- We have learnt about the queueing notation.
- Credits: Jerry Banks book.

Modeling and Simulation

Long-run Measures Of Performance Of Queueing Systems

Overview

- **Here, we will learn about the long-run measures of performance of queueing systems.**

LONG-RUN MEASURES OF PERFORMANCE

- The primary long-run measures of performance of queueing systems are:
- Long-run time-average number of customers in the system (L).
- In the queue (LQ).
- Long-run average time spent in the system (w).

LONG-RUN MEASURES OF PERFORMANCE

- In the queue (wQ) per customer.
- Server utilization.
- A Proportion of time that a server is busy (p).
- Waiting line and mechanism is a system.
- System can also be sub-system of queue system.
- “queue” means waiting line alone.

LONG-RUN MEASURES OF PERFORMANCE

- Other measurements include long-run proportion of customer who is delayed longer than t_o time units.
- Customers turned away due to capacity constraints.
- Waiting line containing more than k_o customers.

LONG-RUN MEASURES OF PERFORMANCE

- G/G/c/N/K is defined.
- Their performance relationships and simulation run.
- The two types of estimators are:
- An ordinary sample average.
- A time-integrated sample average.

Summary

- We have learnt about the long-run measures of performance of queueing systems.
- Credits: Jerry Banks book.

Modeling and Simulation

Properties Of Random Numbers

Overview

- **Here, we are going to learn about properties of random numbers.**

Properties of Random Numbers

- Uniformity and independence are two important statistical properties.
- Each sample must be drawn from continuous uniform distribution between 0 and 1.

$$f(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

Properties of Random Numbers

- The density function and the expected value of R_i is:

$$E(R) = \int_0^1 x\,dx = \frac{x^2}{2}\Big|_0^1 = \frac{1}{2}$$

- And the variance is:

$$V(R) = \int_0^1 x^2\,dx - [E(R)]^2 = \frac{x^3}{3}\Big|_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

Properties of Random Numbers

- Generated by a digital computer.
- Individual computation is expensive so computationally efficient method should be used.
- Routine should be portable for different computers and programming languages.

Properties of Random Numbers

- Routine having a long cycle.
- Replicable random numbers.
- Random numbers should closely approximate the ideal statistical properties.

Properties of Random Numbers

- Techniques to generate random is easy to develop.
- Techniques that produce sequences that are independent and uniformly distributed are difficult.

Summary

- We have learnt about the properties of the random numbers.
- Credits: Jerry Banks book.

Modeling and Simulation

Linear Congruential Method

Overview

- **Here, we will learn about the linear congruential method.**

Linear Congruential Method

- Widely used technique.
- Yields sequences with longer periods.
- Produces the sequence of integers.
- Between zero and $m - 1$.

Linear Congruential Method

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \ldots$$

- x_o = seed.
- a = multiplier.
- c = increment $(c \neq 0)$.
- m = modulus.
- This is called mixed congruential method.
- When c = 0 then known as multiplicative congruential.

Linear Congruential Method

- Interval [0,1] is divided into n classes than expected observation for each interval is N/n .
- N is the total number of observations.
- Observing a value in a particular interval is independent.

Summary

- We have learnt about the linear congruential method.
- Credits: Jerry Banks book.

Modeling and Simulation

Generation Of Pseudo-random Numbers

Overview

- **Here, we will learn about the generation of pseudo-random numbers.**

Generation Of Pseudo-random Numbers

- False number generation.
- Removes the potential of true randomness.
- Set of random numbers can be replicated.
- Producing numbers between 0 and 1 that simulates the properties of uniform distribution and independence.

Generation Of Pseudo-random Numbers

- Some errors or departures are:
- Not uniformly distributed random numbers.
- Not too high or too low variance.
- Generated numbers might not be continuous valued but are discrete valued.

Generation Of Pseudo-random Numbers

- **Dependence:**
- **Autocorrelation between numbers.**
- **Numbers successively higher or lower than adjacent numbers.**
- **Several numbers above the mean followed by the several numbers below the mean.**

Generation Of Pseudo-random Numbers

- **Departures from uniformity and independence for a particular generation scheme often can be detected by tests.**
- **If detected then dropped in the favour of an acceptable generator.**

Summary

- We have learnt about the generation of pseudo-random numbers.
- Credits: Jerry Banks book.

Modeling and Simulation

Tests for Random numbers

Overview

- **Here, we will understand about the Tests for Random numbers.**

Tests for Random numbers

- Two categories:
- Uniformity.

$$H_0: \quad R_i \sim U[0, 1]$$
$$H_1: \quad R_i \not\sim U[0, 1]$$

- H_o denotes numbers distributed uniformly on the interval [0, 1].

Tests for Random numbers

- **Independence.**

$$H_0: \quad R_i \sim \text{independently}$$
$$H_1: \quad R_i \not\sim \text{independently}$$

- **H_o denotes the independent numbers.**

Tests for Random numbers

- A level of significance α must be stated for each test.
- Where α is probability of rejecting the null hypothesis.

$$\alpha = P(\text{reject } H_0 \mid H_0 \text{ true})$$

Tests for Random numbers

- **Software not specifically developed for simulation comes with random number generator pre-installed**

Tests for Random numbers

- **Additional Tests:**
- **Good's serial test, the median-spectrum test, the run's test and a variance heterogeneity test.**

Summary

- Here, we understood about the tests for random numbers.
- Credits: Jerry banks book.

Modeling and Simulation

Random Variate Generation

Overview

- **Here, we will understand about the “Random Variate Generation”.**

Random Variate

- A random variate is a generated variable using uniformly distributed pseudorandom numbers.
- A random variate can be uniformly or nonuniformly distributed.

Random Variate Generation

- **Modeling activities is unpredictable or indeterminate.**

Random Variate Generation

- **Random Variate Generations explain and illustrate some techniques for generating random**

Random Variate Generation

- Sometimes there is no availability of random-variate-generation libraries.
- It is up to the modeller to create an acceptable routine.

Random Variate Generation

- This topic includes
- inverse-transform technique
- Acceptance-rejection technique
- Special properties
- The composition method

Summary

- Here, we understood about Random Variate Generation.
- Credits: Jerry banks book.

Modeling and Simulation

Inverse Transform Technique

Overview

- **Here, we will understand about the “Inverse Transform Technique”.**

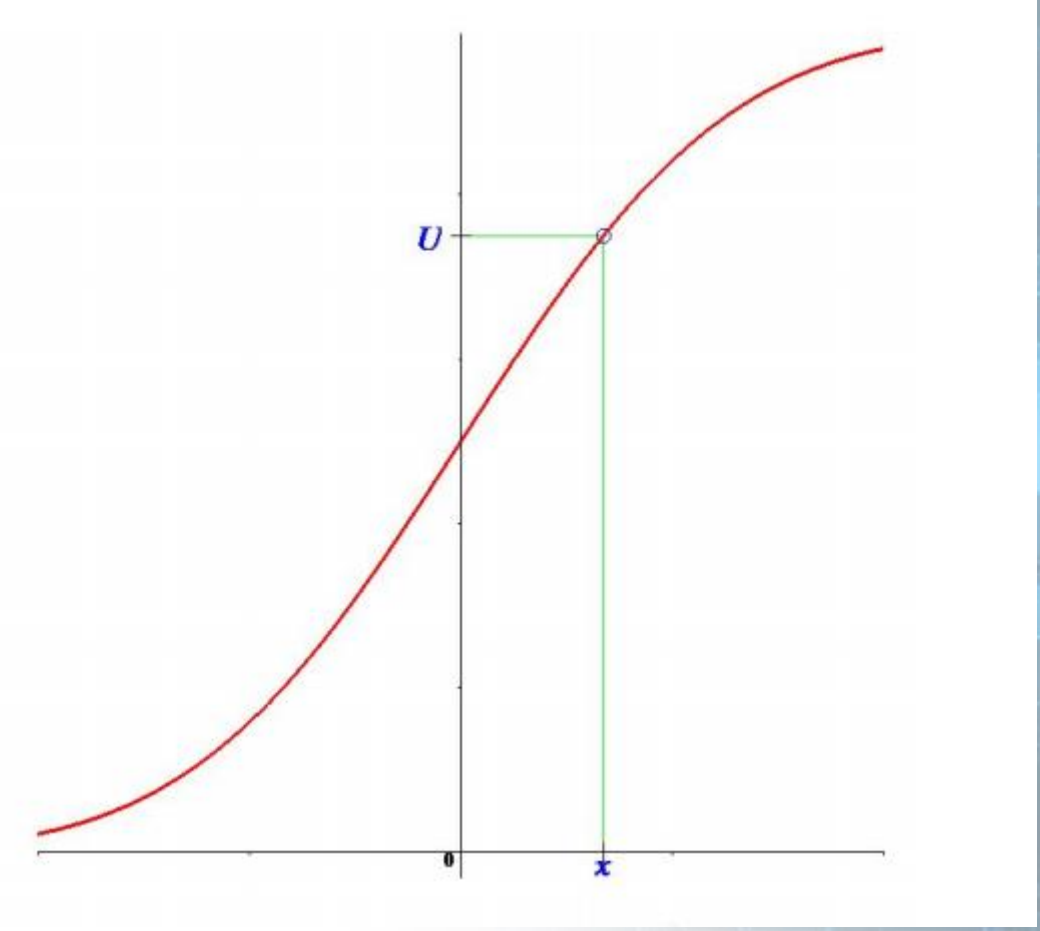
Inverse Transform Technique

- The inverse-transform technique can be used to sample from the exponential.
- The uniform.
- The Weibull.
- The triangular and empirical distributions.

Inverse Transform Technique

- **The basic principle is to find the inverse function of F, F^{-1} such that $FF^{-1} = I$**
- **Where F^{-1} denotes the solution of the equation $r = F(x)$ in terms of γ.**

Inverse Transform Technique



Inverse Transform Technique

- This technique explains:
- Exponential Distribution
- Uniform Distribution
- Weibull Distribution
- Triangular Distribution
- Empirical Continuous Distribution
- Continuous Distribution
- Discrete Distribution

Summary

- Here, we understood about Inverse Transform Technique.
- Credits: Jerry banks book.

Modeling and Simulation

Acceptance Rejection Technique

Overview

- **Here, We will understand “Acceptance Rejection Technique”.**

Acceptance Rejection Technique

- **Acceptance Rejection Technique is a method used to generate random variate using some steps explained next.**

Acceptance Rejection Technique

- **Step 1. Generate a Random Number R.**
- **Step2a. If $R \geq 1/4$ Accept R.**
- **Step2b. If $R \leq 1/4$ Reject R.**
- **If another uniform random variate on [1/4, 1] is needed, repeat the procedure.**

Acceptance Rejection Technique

- **The random variate generated using above methods is indeed uniformly distributed over [1/4, 1].**

Acceptance Rejection Technique

- **Take any** $1/4 \leq a < b \leq 1$ **then**

$$P(a < R \leq b | 1/4 \leq R \leq 1) = \frac{P(a < R \leq b)}{P(1/4 \leq R \leq 1)} = \frac{b-a}{3/4}$$

- **Proves that probability for uniform distribution is on [1/4, 1].**

Acceptance Rejection Technique

- The efficiency of an acceptance-rejection technique rest on reducing the total of rejections.
- Which depends on computer being used.
- The skills of programmer.
- Rejected numbers.

Acceptance Rejection Technique

- The Distributions which operates on acceptance rejection technique are:
- Poisson Distribution
- Nonstationary Poisson process
- Gamma Distribution.

Summary

- Here, we understood about the Acceptance Rejection Technique.
- Credits, Jerry Banks book.

Modeling and Simulation

Other topics in Randomness

Overview

- **Here, we will discuss about “Other topics in Randomness”.**

Special Properties

- **Special Properties are variate generation techniques based on a particular feature of probability distributions.**

Direct Transformation

- Direct transformation is used for the normal and lognormal distributions.
- The cdf is given by:

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt, \qquad -\infty < x < \infty$$

- Box and Muller made this method and is easy to program in scientific programming languages.

Convolution Method

- **The probability distribution of a sum of two or more independent random variables is called convolution method.**

Convolution Method

- This method uses two random variables.
- Sum them and generate a new random variable.
- This technique can be applied on Erlang and Binomial variates.

More Special Properties

- The other relationship among distributions include relation between gamma and beta distribution.
- We can generate beta variates, using two gamma variates with acceptance-rejection technique.

Summary

- Here, We understood about the Special Properties, Direct transformation, convolution method and other special properties.
- Credits: Jerry banks book.

Modeling and Simulation

Data Collection

Overview

- **Here, we will discuss about “Data Collection”.**

Data Collection

- **The relation between construction of the model and the collection of needed input data is constant.**

Data Collection

- The required data elements changes with complexity of the model.
- Time taking nature of data collection.

Data Collection

- **Which type of data is to be collected is dictated by the objectives of study.**

Summary

- Here, We understood about the Data Collection.
- Credits: Jerry banks book.

Modeling and Simulation

Identifying the Distribution with Data

Overview

- **Here, we will understand About “Identifying the Distribution with Data”.**

Identifying the Distribution with Data

- **Data distribution are graphical methods.**

Identifying the Distribution with Data

- These methods are used for selecting input distribution families.
- The specific distribution is often specified by estimating its parameters.

Summary

- We understood about Data Distribution and how data parameters are selected.
- Credits, Jerry Banks book.

Modeling and Simulation

Histograms

Overview

- **Here, we will understand about the “Histograms”.**

Histograms

- **Histograms is a powerful technique for identifying how a distribution is shaped.**
- **Histogram is also known as frequency distribution.**

Histograms

- **There are some methods for constructing a histogram.**

Histograms

- 1. Divide the data ranges into intervals of equal width.
- 2. Label the horizontal axis to follow the nominated intervals.
- 3. Find the frequency of occurrences inside each intervals.

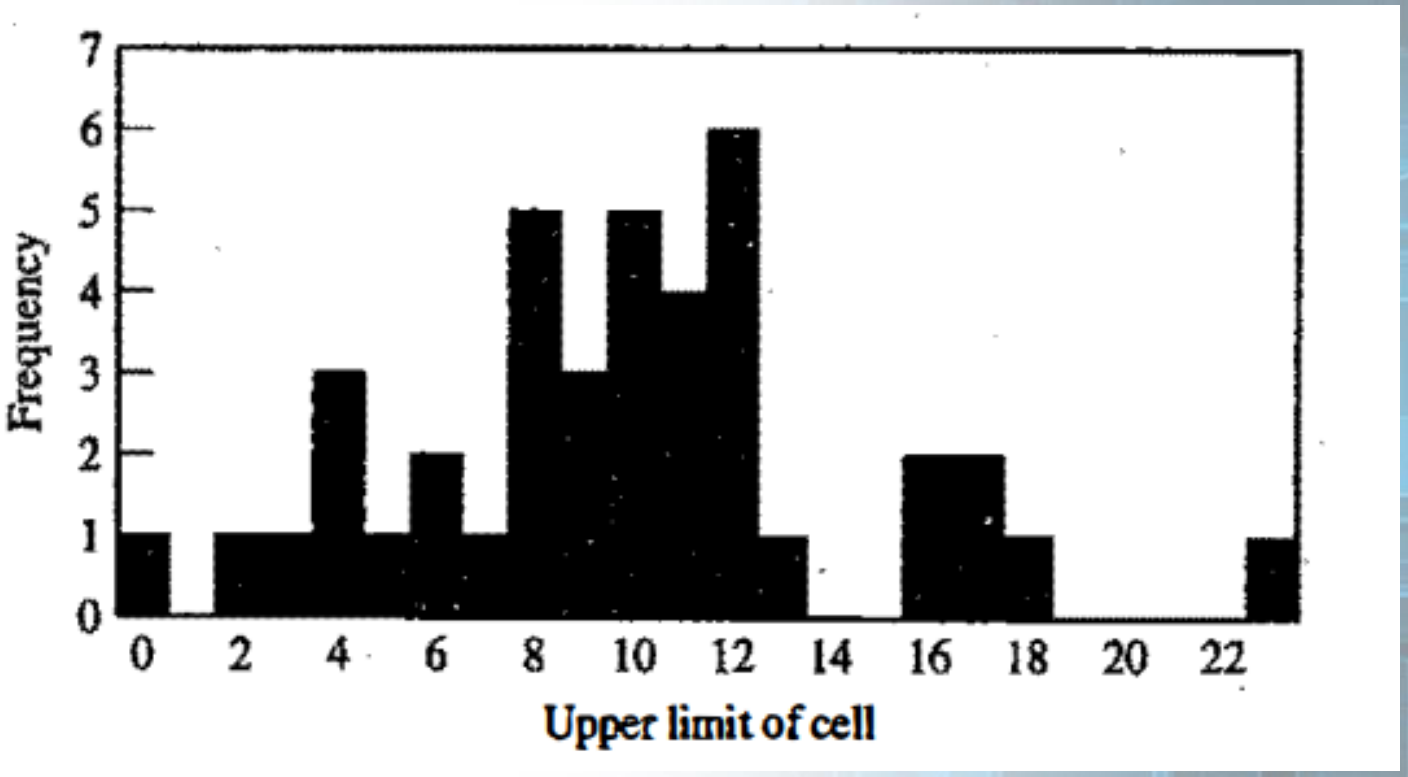
Histograms

- **4. Label the vertical axis to plot total occurrences for every interval.**
- **5. Plot the frequencies on the vertical axis.**

Histograms

- The number of class intervals is decided by the number of observations.
- Amount of scatter in the data.

Histograms


Frequency
0
1
2
3
4
5
6
7
0
2
4
6
8
10
12
14
16
18
20
22
Upper limit of cell

Summary

- Here, we understood about the “Histograms”.
- Credits: jerry banks book.

Modeling and Simulation

Selecting the Family of Distributions

Overview

- **Here, we will understand about the “Selecting the Family of Distributions”.**

Selecting the Family of Distributions

- **The selection of a family of distributions is based on the context for what it is being investigated.**
- **The shape of the histogram.**

Selecting the Family of Distributions

- **The Distributions which are frequently encountered and easy to analyze includes exponential, normal and poison distribution.**
- **Whereas difficult to analyze includes the beta, gamma and Weibull distributions.**

Selecting the Family of Distributions

- Out of many, Here are some examples of distributions which were created on the physical basis.
- Binomial
- Negative Binomial
- Poisson
- Normal
- Lognormal

Selecting the Family of Distributions

- An input Model is an estimate of reality.
- There is no “true” distribution for any stochastic input process.

Summary

- Here, we understood about the Family of distributions, its types and input model.
- Credits: Jerry banks book.

Modeling and Simulation

Quantile-Quantile Plot

Overview

- **Here, we will understand about the “Quantile-Quantile Plot”.**

Quantile-Quantile Plot

- **A Quantile-Quantile is a graphical method for comparison of two probability distributions by plotting their qualities against each other.**

Quantile-Quantile Plot

- **If $F(\gamma) = P(X \leq \gamma) = q$ for $0 < q < 1$**
- **Here x is random variable with cdf F.**
- ***Q-quantile of X is value of γ .***
- ***We write $\gamma = F^{-1}(q)$ when F has an inverse.***

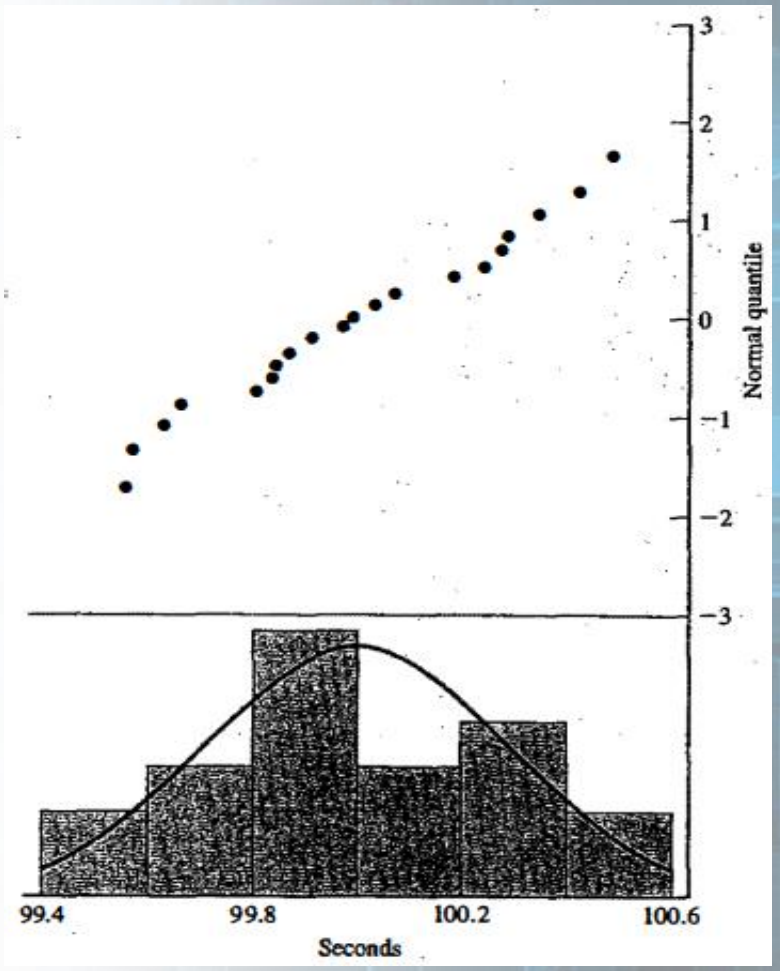
Quantile-Quantile Plot

- If $\{x_i, \boldsymbol{i} = 1,2,3 \cdot \cdots n\}$ and $\{\boldsymbol{y_j}, \boldsymbol{j} = 1,2,3 \cdot \cdots n\}$
- Where $y_1 \leq y_2 \leq \cdots \leq y_n$
- Let $\boldsymbol{j}$ denote the ranking of order
- $j = 1$for smallest
- $j = \boldsymbol{n}$ for largest

Quantile-Quantile Plot

- **Then**
- $y_j \approx F^{-1}\left(\frac{j-\frac{1}{2}}{n}\right)$

Quantile-Quantile Plot



Summary

- Here, we understood about the “Quantile-Quantile Plot”.
- Credits: jerry banks book.

Modeling and Simulation

PARAMETER ESTIMATION

Overview

- **Here, we will learn about the parameter estimation.**

Parameter Estimation

- **After selecting the family of distribution, the next step is to estimate the parameters of the distribution.**

Parameter Estimation

- **Many useful distribution estimators are described here.**

Parameter Estimation

- **Software packages integrated into simulation languages are available.**

Parameter Estimation

- We have learnt about the parameter distribution.
- Credits: Jerry Banks book.

Modeling and Simulation

Preliminary Statistics: Sample Mean and Sample Variance.

Overview

- **Here, we will learn about the preliminary statistics: sample mean and sample variance.**

Preliminary Statistics

- **To estimate the parameters of a hypothesized distribution.**

Preliminary Statistics

- When discrete and continuous raw data is available then the following equation is used:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

- $\bar{x}$ = sample mean.
- $x_1, x_2, \ldots, x_n$ = observation in a sample of size n.

Preliminary Statistics

- **The sample variance s^2 is defined as:**

$$S^2 = \frac{\sum_{i=1}^{n} X_i^2 - n\bar{X}^2}{n-1}$$

Preliminary Statistics

- When discrete or continuous raw data are available following modified equations are used:

$$\bar{X} = \frac{\sum_{j=1}^{k} f_j X_j}{n}$$

- Where $\bar{x}$ is the sample mean.

Preliminary Statistics

- And the sample variance is :

$$S^2 = \frac{\sum_{j=1}^{k} f_j X_j^2 - n\bar{X}^2}{n-1}$$

- Here k is the number of distinct values of X.
- F_j is the observed frequency of the value X_j of X.

Preliminary Statistics

- When data is discrete or continuous and have been placed in class intervals then the sample mean and the sample variance are approximated from the following equations:

$$\bar{X} \doteq \frac{\sum_{j=1}^{c} f_j m_j}{n}$$

Preliminary Statistics

- For the sample variance the following equation is used.

$$S^2 \doteq \frac{\sum_{j=1}^{c} f_j m_j^2 - n\bar{X}^2}{n-1}$$

- f_j observed frequency of j_{th} class interval.
- m_j midpoint of j_{th} interval.
- c = number of intervals.

Summary

- We have learnt about the preliminary statistics: sample mean and sample variance.
- Credits: Jerry Banks book.

Modeling and Simulation

Evolution of Computer Systems Architectures

Overview

- **Here, we will learn about the evolution of computer systems architectures**

Evolution of Computer Systems Architectures

- ENIAC was the first computer system with no operating system and was large in size. They used vacuum tubes.
- Early computer doesn't have a good operating system, databases.
- They doesn't have any high-level programming language for simplifying program.

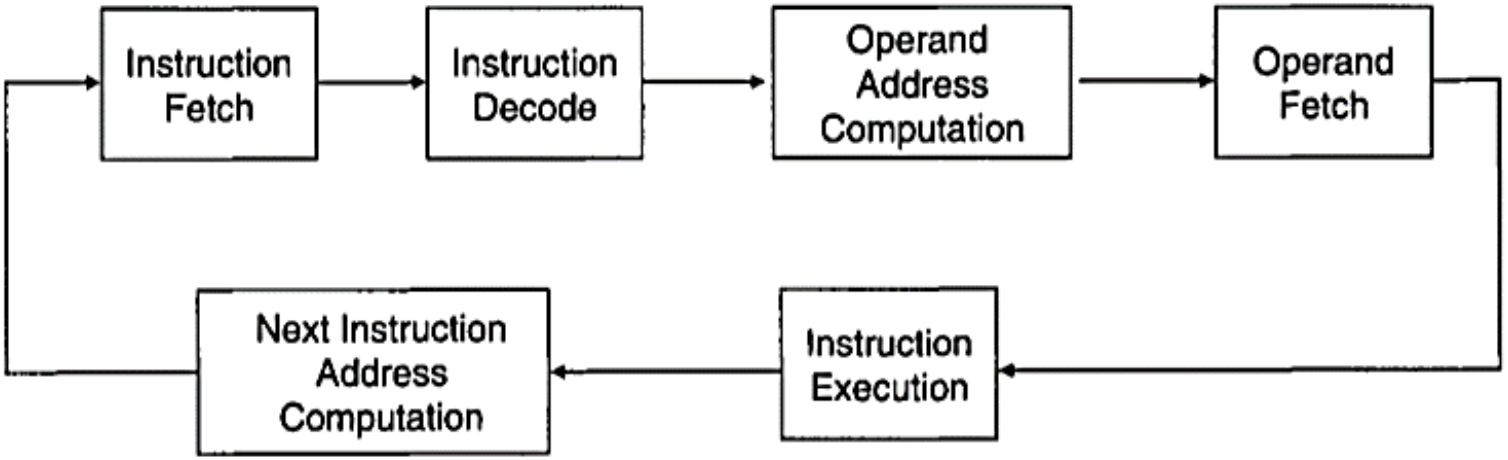
Systems Architecture

- Central Processing Unit.
- Graphics Processing Unit.
- Temporary memory.
- Permanent Memory.
- Input unit.
- Output unit.
- Control element.
- Disk Drives.

CPU Architectures

- Central processing unit.
- CU fetch functions.
- ALU performs functions.
- Registers to store the operands.
- L1, L2, L3 Cache.
- CPU Execution cycle.

Instruction architectures



Instructions and Memory Architectures

- Reduced instruction set computer (RISC).
- Highly optimized instructions.
- Complex instruction set computer (CISC).
- Memory address register.
- Memory data register.

Secondary Storage and Peripheral Device Architecture

- I/O devices control secondary storage.
- Magnetic tape drives.
- Magnetic disk drives.
- Compact optical disk drives.
- Disk jukeboxes.
- High density data storage.

Network Architecture

- Share information and processing capacity in real life.
- Another path to send or receive data.
- Central switching element. Central storage repository.
- Connected using intelligent interface units.

Summary

- We have learnt about the evolution of computer systems architectures.
- Credits: Paul Fortier book.

Modeling and Simulation

Evolution of Database Systems

Overview

- **Here, we will learn about the evolution of database systems.**

Evolution of Database Systems

- Powerful tools to manage information.
- Early computers lag in online data storage.
- System architects have to reply on external storage.
- Initial storage was constructed using simple direct addressing.

Evolution of Database Systems

- File system used coarse semantic means.
- Information Centralization.
- Security.
- Integrity.
- File systems added finer-grained definitions of stored information.

Evolution of Database Systems

- Relational database system was introduced.
- Database must be entered at a specific entry point.
- Must traverse the path leading to the specific item.
- An issue for network is its legacy.

Evolution of Database Systems

- Database system's demise began with development.
- Publication of Codd's relational database model.
- Information can be formed into tables called relations.
- Have to use calculus and algebra to operate.

Evolution of Database Systems

- Concurrent access was not present in early network-based databases.
- The serializability theory and concurrency control led to further improvements in database technology.
- “ACID” properties.

Evolution of Database Systems

- Object-oriented database was developed.
- For application developers to store data and data types efficiently.
- But did not possess some fundamental concepts.

Evolution of Database Systems

- Object relational databases.
- Embraced by the U.S and international venders.
- Next change will be in the area of transactions and transaction processing.
- ACID properties may be altered.

Summary

- We have learnt about the evolution of database systems.
- Credits: Paul Fortier book.

Modeling and Simulation

Evolution of Operating Systems

Overview

- **Here, we will learn about the evolution of operating systems.**

Evolution of Operating Systems

- Operating system is computer's software, firmware, and hardware.
- Interacts at low-level.
- Manage sharing of computer's resources.
- The most privileged of software elements on the system.
- Requires hardware.

Evolution of Operating Systems

- Early coders load the software in a location then make hardware to start processing.
- No automated means to switch between different jobs.
- Monitor or batch operating system provided the solution to transition between processing jobs.

Services of Operating System

- Process and Hardware management.
- Memory management.
- I/O management.
- Inter-process synchronization and communications.
- Resource allocation.
- Protection of system and user resources.

Evolution of Operating Systems

- interrupt service routine checks everything and determines what to perform next.
- Efficient use of computing facilities.
- Convenient interface to user by hiding the details of machine.
- Transparent use of computer resources.

Evolution of Operating Systems

- Improved architectures and instruction execution schemes.
- Improve the performance of the processor.
- Not to depend on a single processor
- Add entire new processors.

Evolution of Operating Systems

- Distributed processors.
- Processors operating system is the single global operating system.
- In one case the entire operating system can be replicated on each site.
- Each processor has a specific function.

Evolution of Operating Systems

- New concepts are still being examined.
- Client/server processing.
- Multiprocessing operating system forum.

Summary

- We have learnt about the evolution of operating systems.
- Credits: Paul Fortier book.

Modeling and Simulation

Evolution of Computer Networks

Overview

- **Here, we will learn about the evolution of computer networks**

Evolution of Computer Networks

- The term network can mean many things.
- In computer it refers to combination of interconnected resources.
- ARPANET was an early computer network.
- A loose coalition of devices that share information.

Evolution of Computer Networks

- All this evolved into internet.
- ARPANET provided early research into communication protocols.
- Greater availability and access to a wider range of devices.
- As more applications were added.

Evolution of Computer Networks

- Information access and manipulation took on greater emphasis.
- As the networks improve new applications arose.
- Remote job execution was added with information exchange.
- But still real-time was not provided.

Evolution of Computer Networks

- They had to develop more advance protocols and network operating systems.
- It was time consuming.
- Local area network was introduced in 1970s.
- LAN became more widely available.

Evolution of Computer Networks

- LAN designs have been produced to fit an extremely wide spectrum of user requirements.
- LAN is widely used in all aspects of society.
- LAN had rapid growth in 1990s and early 2000s.
- LAN can connect a device from anywhere.

Evolution of Computer Networks

- LAN is used to link factory robots.
- LAN provide sensor data, control data and feedback.
- Walt Disney World used LAN for different services.
- Large banks also use LAN for interconnection of their local sites.

Evolution of Computer Networks

- Wireless technology has improved.
- Began in 1970s with Aloha net as foundation.
- Wireless phone network development has opened the door for computer networks.
- More development will be made to provide more applications.

Summary

- We have learnt about the evolution of computer networks.
- Credits: Paul Fortier book.

Modeling and Simulation

Need for Performance Evaluation

Overview

- **Here, we will learn about the need for performance evaluation**

Need for Performance Evaluation

- Selecting a specific computer architecture for an application.
- An operating system.
- A database system.
- A wide area or local area network system.
- If not used properly then decrease productivity.

Need for Performance Evaluation

- Operating system can increase the concurrency access.
- Or can block access.
- A database can provide efficiently sharing information among many applications.
- Or blocking of information by dropping data availably.

Need for Performance Evaluation

- LAN provides means for streamline information processing and eliminate redundancies.
- But it may also deter users from logging on because of link or protocol problems.
- We need to select the computer system mapped to the needs.

Need for Performance Evaluation

- A computer system can be very simple with simple processor etc.
- A computer can be highly elaborate with multiple processors etc.
- A purchaser must decide according to motherboard its using.
- While purchasing the system all this must be kept in mind.

Summary

- We have learnt about the Need for performance evaluation.
- Credits: Paul Fortier book.

Modeling and Simulation

Role of Performance Evaluation in Computer Engineering

Overview

- **Here, we will learn about role of performance evaluation in computer engineering.**

Role of Performance Evaluation in Computer Engineering

- Client/server architectures.
- These systems are from Government, industry and academia.
- There is constant demand for improved reliability and availability.

Role of Performance Evaluation in Computer Engineering

- Present systems provide more features.
- Sharing of resources on global scale.
- Ability to bring computing power to the user.

Role of Performance Evaluation in Computer Engineering

- **The optimal design or selection of computer system is therefore very important.**

Summary

- We have learnt about the role of performance evaluation in computer engineering.
- Credits: Paul Fortier book.

Modeling and Simulation

Overview of Performance Evaluation Methods

Overview

- **Here, we will learn about the overview of performance evaluation methods.**

Overview of Performance Evaluation Methods

- Models provide a tool to define a system.
- Define its problems in a concise fashion.
- This provide an environment to study a system before its existence.
- Developed on the base of theoretical laws and principles.

Overview of Performance Evaluation Methods

- Modeling is better if:
- Physical laws are available.
- Pictorial representation is possible.
- Input/output elements are manageable.
- But we don’t have clear physical laws.

Overview of Performance Evaluation Methods

- Faithful model of a system must be constructed.
- Model a slice of the real-world system.
- Which element is important to understand in faithful system.
- We can say its intuition.

Summary

- We have learnt about the overview of performance evaluation methods.
- Credits: Paul Fortier book.

Modeling and Simulation

Performance Metrics and Evaluation Criteria

Overview

- **Here, we will learn about performance metrics and evaluation criteria.**

Performance Metrics and Evaluation Criteria

- Performance metrics and evaluation criteria is required for systems.
- User must follow a methodology.
- The motivations.
- The environment.
- Technological boundaries.

Performance Metrics and Evaluation Criteria

- Must understand the usage of the system.
- Needs and uses must be defined.
- Must compile a wish list of all potential uses.

Performance Metrics and Evaluation Criteria

- User must know processing requirements.
- Communication transfer.
- Management requirements.
- Define or know why we need something at first priority.

Performance Metrics and Evaluation Criteria

- Checking in which environment the system fits.
- Checking in which technological aspect it fits.
- Connecting to diverse set of company assets.
- Link planned new resources.

Performance Metrics and Evaluation Criteria

- Need large volume of data for system requirements.
- How this data assist in the selection?
- Collection of data can be divided in quantitative and qualitative.

Performance Metrics and Evaluation Criteria

- **Specific performance measures can be derived from one set of data.**
- **Subjective measures can be derived from another.**

Summary

- We have learnt about the performance metrics and evaluation criteria.
- Credits: Paul Fortier book.

Modeling and Simulation

Comparison of Two System Designs

Overview

- **Here, we will understand about the “Comparison of Two System Designs”.**

Comparison of Two System Designs

- **A simulation analyst can compare two possible configurations.**
- **Using supply chain inventory system there ordering policies will be compared.**

Comparison of Two System Designs

- **Replications is used to analyze the output data.**
- **The mean performance measure for system 'I' will be denoted by $\theta_i = (\boldsymbol{i} = 1,2)$.**

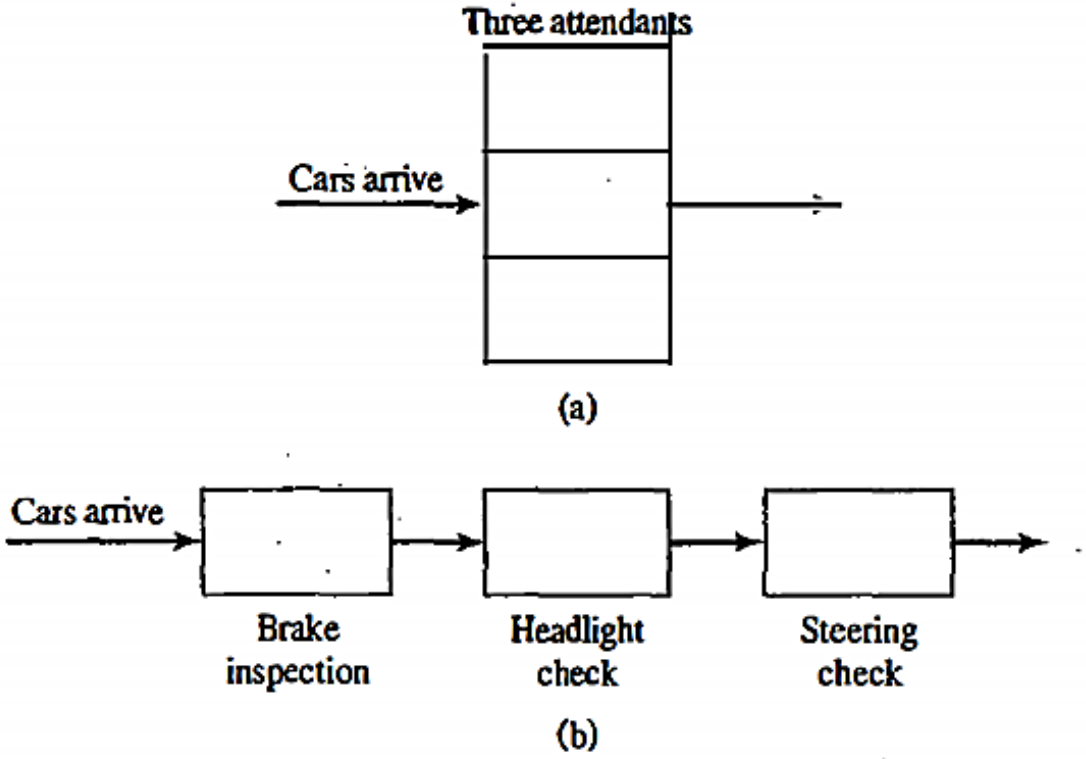
Comparison of Two System Designs

- The goal of the simulation experiments is to obtain point.
- Interval estimates of the difference in mean performance.
- Namely $(\theta_1 - \theta_2)$.

Comparison of Two System Designs

- **Inverse Transform Technique**
- **Replications is used to analyze the output data.**
- **The mean performance measure for system 'I' will be denoted by** $\theta_i = (\boldsymbol{i} = 1,2)$.

Comparison of Two System Designs



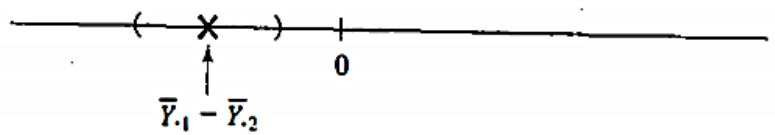
Vehicle Safety Inspection

Comparison of Two System Designs

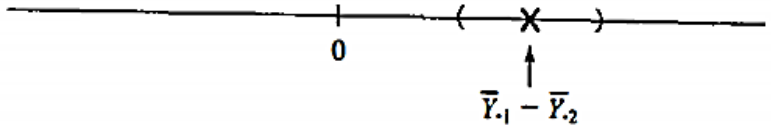
- **In comparing two systems, number of replications R_i to be made and run length $T_E^{(i)}$ for each model $(\boldsymbol{i} = 1,2)$ is decided.**

Comparison of Two System Designs

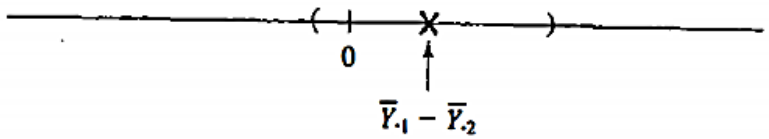
System	Replication 1	2	...	R_i	Sample Mean	Sample Variance
1	Y_{11}	Y_{21}	...	$Y_{R1}1$	$\overline{Y}_{\cdot 1}$	S_1^2
2	Y_{12}	Y_{22}	...	$Y_{R2}2$	$\overline{Y}_{\cdot 2}$	S_2^2



(a)



(b)



(c)

Summary

- **Here, we understood about Comparing two system designs.**
- **Credits: Jerry banks book.**

Modeling and Simulation

Independent Sampling with Equal Variances

Overview

- **Here, We will understand “Independent Sampling with Equal Variances”.**

Independent Sampling with Equal Variances

- **Independent Sampling refers to different and independent random numbers streams to simulate the two systems.**

Independent Sampling with Equal Variances

- **Acceptance-Rejection**
- **The observations of simulated system 1, $\{Y_{r_1}, r = 1, \cdots, R_1\}$ are all statistically independent of simulated system 2, , $\{Y_{r_2}, r = 1, \cdots, R_2\}$.**
- **Y_i is given by:**

$$V(\bar{Y}_i) = \frac{V(Y_{ri})}{R_i} = \frac{\sigma_i^2}{R_i}, \quad i = 1, 2$$

Independent Sampling with Equal Variances

- **Where $\overrightarrow{Y_1}$ and $\overrightarrow{Y_2}$ are statistically different as:**

$$V(\bar{Y}_{.1} - \bar{Y}_{.2}) = V(\bar{Y}_{.1}) + V(\bar{Y}_{.2}) = \frac{\sigma_1^2}{R_1} + \frac{\sigma_2^2}{R_2}$$

- **It implies two variances are equal (but unknown in value) $\sigma_1^2 = \sigma_2^2$.**

Independent Sampling with Equal Variances

- If two-sample-t confidence interval approach is used. The point estimate of the mean difference is: $\overrightarrow{Y_1}$ - $\overrightarrow{Y_2}$
- Where $\overrightarrow{Y_i}$ is given as:

$$S_i^2 = \frac{1}{R_i - 1}\sum_{r=1}^{R_i}(Y_{ri} - \overline{Y}_{\cdot i})^2$$

$$= \frac{1}{R_i - 1}\left(\sum_{r=1}^{R_i} Y_{ri}^2 - R_i\overline{Y}_{\cdot i}^2\right)$$

Independent Sampling with Equal Variances

- Here, S_i^2 is an estimator of variance σ_i^2 . Assuming $\sigma_1^2 = \sigma_2^2 = \sigma^2$ a pool estimate of σ^2 is obtained as:

$$S_p^2 = \frac{(R_1 - 1)S_1^2 + (R_2 - 1)S_2^2}{R_1 + R_2 - 2}$$

- Where $v = R_1 + R_2 - 2$

Independent Sampling with Equal Variances

- **The c.i. $\theta_1 - \theta_2$ for with the standard error is computed by:**

$$s.e.(\overline{Y}_{\cdot 1} - \overline{Y}_{\cdot 2}) = S_p \sqrt{\frac{1}{R_1} + \frac{1}{R_2}}$$

- **The standard error is an estimate of the standard deviation of the point given as:**

$\sigma\sqrt{1/R_1 + 1/R_2}$.

Summary

- Here, we understood about the Independent Sampling.
- Credits, Jerry Banks book.

Modeling and Simulation

Independent Sampling with Unequal Variances

Overview

- **Here, we will discuss about “Independent Sampling with Unequal Variances”.**

Independent Sampling with Unequal Variances

- **If the assumption of equal variances cannot safely be made, an approximate $100(1-\alpha)\%$ for $\theta_1 - \theta_2$ can be computed as:**

$$\text{s.e.}(\overline{Y}_{.1} - \overline{Y}_{.2}) = \sqrt{\frac{S_1^2}{R_1} + \frac{S_2^2}{R_2}}$$

Independent Sampling with Unequal Variances

- **Degrees of freedom, v can be as:**

$$v \approx \frac{(S_1^2/R_1 + S_2^2/R_2)^2}{[(S_1^2/R_1)^2/(R_1-1)] + [(S_2^2/R_2)^2/(R_2-1)]}$$

- **A minimum number of replications $R_1 \geq 6$ and $R_2 \geq 6$ is recommended for this procedure.**

Summary

- **Here, We understood about Independent Sampling with Unequal Variances.**
- **Credits: Jerry banks book.**

Modeling and Simulation

Common Random Numbers

Overview

- **Here, we will understand About “Common Random Numbers”.**

Common Random Numbers

- **CRN means, for each replication, the same random numbers are used to simulate both system.**
- **Therfore, R1 and R2 must be equal, say R1 = R2= R.**

Common Random Numbers

- The purpose of CRN is to induce positive correlation between y_1 ***and*** y_2.
- A variance reduction in point estimator of mean difference Type equation here. is given by:

$$V(\bar{Y}_1 - \bar{Y}_2) = V(\bar{Y}_1) + V(\bar{Y}_2) - 2\mathrm{cov}(\bar{Y}_1, \bar{Y}_2)$$
$$= \frac{\sigma_1^2}{R} + \frac{\sigma_2^2}{R} - \frac{2\rho_{12}\sigma_1\sigma_2}{R}$$

Common Random Numbers

- **Compare variance of $\overrightarrow{Y_1} - \overrightarrow{Y_2}$, call it V_{crn} given as :**

$$V_{CRN} = V_{IND} - \frac{2\rho_{12}\sigma_1\sigma_2}{R}$$

- **Which implies:**

$$V_{CRN} < V_{IND}$$

Common Random Numbers

- To compute a 100(1 - a)% c.i compute the differences:

$$D_r = Y_{r1} - Y_{r2}$$

- Compute Sample mean difference:

$$\bar{\bar{D}} = \frac{1}{R}\sum_{r=1}^{R} D_r$$

Common Random Numbers

- The sample variance of the difference $\{D_r\}$ is compute as:

$$S_D^2 = \frac{1}{R-1}\sum_{r=1}^{R}(D_r - \bar{D})^2$$
$$= \frac{1}{R-1}\left(\sum_{r=1}^{R} D_r^2 - R\bar{D}^2\right)$$

- Where standard error of $\overrightarrow{Y_1} - \overrightarrow{Y_2}$ is estimated by:

$$\text{s.e.}(\bar{D}) = \text{s.e.}(\bar{Y}_1 - \bar{Y}_2) = \frac{S_D}{\sqrt{R}}$$

Common Random Numbers

- Implementation of common random number.

1. Dedicate a random-number stream to a specific purpose.
2. For system with external arrivals, dedicate one random stream to one of arrival and their attributes.

Common Random Numbers

3. For systems having an entity performing given activities in a cyclic fashion, assign a random-number stream to this entity.
4. If synchronization is not possible use independent steams of random numbers for this subset.

Summary

- We understand about Common Random Numbers.
- Credits, Jerry Banks book.

Modeling and Simulation

Confidence Intervals with Specified Precision

Overview

- **Here, we will understand about the “Confidence Intervals with Specified Precision”.**

Confidence Intervals with Specified Precision

- **Confidence intervals for the difference between two systems' performance can be achieved in an parallel way.**

Confidence Intervals with Specified Precision

- **If the goal is to obtain error in our estimate $\theta_1 - \theta_2$ to be less than $\pm \varepsilon$ then number of replications R should be:**

$$H = t_{\alpha/2,\nu} \text{s.e.}(\bar{Y}_{.1} - \bar{Y}_{.2}) \leq \epsilon$$

Confidence Intervals with Specified Precision

- Using $R_0 \geq 2$ we obtain an initial estimate.
- We solve total number of replication $R \geq R_0$ to achieve half-length criterion.
- We make replications $R \geq R_0$ for each system to compute confidence interval and half-length criterion.

Summary

- Here we understand about Confidence Intervals with Specified Precision.
- Credits: Uri Wilensky book

Modeling and Simulation

Comparison of Several System Designs

Overview

- **Here, we will understand about the “COMPARISON OF SEVERAL SYSTEM DESIGNS”.**

COMPARISON OF SEVERAL SYSTEM DESIGNS

- To compare "K" alterative system designs.
- Many different statistical procedures have been developed.
- These are used to analyze simulation data and to draw statistically sound interfaces about parameters θ_i.

COMPARISON OF SEVERAL SYSTEM DESIGNS

- **These procedures are classified as:**
- **Fixed-sample size.**
- **Sequential sampling.**

COMPARISON OF SEVERAL SYSTEM DESIGNS

- **Fixed Sample Size**
- **A predetermined sample size is used to draw inferences via hypothesis tests or confidence intervals.**
- **Example:**
- **The interval estimation of a mean performance measure.**

COMPARISON OF SEVERAL SYSTEM DESIGNS

- Advantages
- An easily estimated or known cost in terms of computer time before running the experiments.
- Can be used when :
- Computer time is limited.
- A pilot study is being conducted.

COMPARISON OF SEVERAL SYSTEM DESIGNS

- **Disadvantages**
- **A strong Conclusion could be impossible.**
- **A hypothesis test may lead to a failure to reject the null hypothesis.**

COMPARISON OF SEVERAL SYSTEM DESIGNS

- **Sequential Sampling**
- **In sequential sampling more and more data are collected until an estimator with a pre-specified precision is achieved.**
- **In two stage procedure, an initial sample estimates additional observations needed to draw conclusions with a specified precision.**

COMPARISON OF SEVERAL SYSTEM DESIGNS

- Goal of simulation analyst
- Estimation of each parameter, θ_i.
- Comparison of each performance measure.
- All pair wise comparisons.
- selection of the best θ_i.

Summary

- Here, we understood about the “COMPARISON OF SERAL SYSTEM DESIGNS”.
- Credits: jerry banks book.

Modeling and Simulation

Bonferroni Approach to Multiple Comparisons

Overview

- **Here, we will understand about the “Bonferroni Approach to Multiple Comparisons”.**

Bonferroni Approach to Multiple Comparisons

- If C confidence intervals are computed and that the it_h interval has confidence coefficient $1 - \alpha_i$.
- The Bonferroni inequality states that:

$$P(\text{all statements } S_i \text{ are true}, i = 1, \ldots, C) \geq 1 - \sum_{j=1}^{C} \alpha_j = 1 - \alpha_\varepsilon$$

Bonferroni Approach to Multiple Comparisons

- Where $\alpha_E = \sum_{j=1}^{C} \alpha_j$ is called overall error probability.
- This is equivalent to:

$$P(\text{one or more of the } C \text{ confidence intervals does not contain the parameter being estimated}) \leq \alpha_E$$

- Here, a^E provides the upper bound.

Bonferroni Approach to Multiple Comparisons

- **A major advantage of Bonferroni approach is that runs with independent sampling and or common random numbers.**

Bonferroni Approach to Multiple Comparisons

- **The major disadvantage of the Bonferroni approach in making a large number of comparisons is the increased width of each individual interval.**
- **The width of a c.i. is a measure of the precision of the estimate.**

Bonferroni Approach to Multiple Comparisons

- **3 possible ways of using the Bonferroni Inequality when comparing K alterative system designs:**
- **1. Construct a** $100(1-\alpha_i)\%$ **c.i. for parameter** θ_i **, where number of intervals is C = K.**

Bonferroni Approach to Multiple Comparisons

- **2. Compare all designs to one specific design that is:**
- **Construct a** $100(1-\alpha_i)\%$ **c.i. for** $\theta_i - \theta_1$.
- **Here, the number of intervals is C = K – 1.**

Bonferroni Approach to Multiple Comparisons

- **3. Compare all designs to each other. That is :**
- **Construct a** $100(1-\alpha_i)\%$ **c.i. for** $\theta_i - \theta_j$.
- **With K designs, the number of intervals is computed as**
  C = K (K – 1) / 2.

Summary

- Here, we understood about Bonferroni Approach.
- Credits: Jerry banks book.

Modeling and Simulation

Bonferroni Approach to Selecting the Best

Overview

- **Here, We will understand “Bonferroni Approach to Selecting the Best”.**

Bonferroni Approach to Selecting the Best

- Gross Level
- If there are K system designs, and the i_{TH} design has expected performance θ_i.
- Then , we're interested in best system.
- Where "best" means maximum expected performance.

Bonferroni Approach to Selecting the Best

- Refined
- At a refined level, we are also interesting in knowing how much better the best is relative to alternatives.
- It helps us in choosing a inferior system not deficient by much.

Bonferroni Approach to Selecting the Best

- Difference
- If system design i is the best, then $\theta_i - \max_{j \neq i} \theta_j$ is equal to the difference in performance between the best and the second best.
- If system design i is not the best then, $\theta_i - \max_{j \neq i} \theta_j$ is equal to the difference between system i and the best.

Bonferroni Approach to Selecting the Best

- i^* denote index of the best system.
- Smaller the difference $\theta_i - \max_{j \neq i} \theta_j$ is, the more certain we find the best system.
- If the difference between system i^* and the other is significant, then we try to achieve i^* with high probability.

Bonferroni Approach to Selecting the Best

- The best system probability should be at least :

$1 - \alpha$ whenever $\theta_{i^*} - \max_{j \neq i^*} \theta_j \geq \epsilon$

- We select best of one of near best if multiple system are within ε of the best.

Bonferroni Approach to Selecting the Best

- **Achievement**
- **This procedure achieves the desired probability and also forms** $100(1-a)\%$ **confidence intervals for**
  $\theta_i - \max_{j \neq i} \theta_j$

Summary

- Here, we understood about Bonferroni Approach to Selecting the Best.
- Credits, Jerry Banks book.

Modeling and Simulation

Bonferroni Approach to Screening

Overview

- **Here, we will discuss about “Bonferroni Approach to Screening”.**

Bonferroni Approach to Screening

- When there exist multiple systems and two stage procedure is not possible.
- Then, divide set of systems into the best.
- This is achieved by using screening or subset selection procedure.

Bonferroni Approach to Screening

- Independent sampling with unequal variances
- This promises that stayed subset contains best system.
- With probability $\geq 1 - a$.
- When the data is distributed normally, by independent sampling or CRN is used.

Bonferroni Approach to Screening

- This subset may contain all K systems, only one or some.
- It depends on the number of replications and sample variances.

Bonferroni Approach to Screening

- Procedure
- Specify correct probability selection and sample size.
- Make R replications of the system.
- Calculate the sample and sample variance of the difference.

Bonferroni Approach to Screening

- If bigger is better then retain selected subset system.
- If smaller is better then retain selected subset system.
- All design not retained can be eliminated from further cosiderations.

Summary

- Here, We understood about Bonferroni Approach to Screening.
- Credits: Jerry banks book.

Modeling and Simulation

Metamodeling

Overview

- **Here, we will understand About “Metamodeling”.**

Metamodeling

- **Metamodeling is the process of generating metamodels.**
- **Where metamodel refers to model of a model.**

Metamodeling

- There is a simulation output variable Y.
- It is related to k independent variables.
- Dependent variable Y is random variable.
- Independent variable is design variable.

Metamodeling

- Relation between independent and dependent variable is represented by simulation model.
- Metamodel simplifies this relationship by simpler mathematical function.

Metamodeling

- This relationship is represented by as function $Y = f(x)$.
- Most of the time relationship is unknown.
- Function is created using unknown parameter.
- Regression analysis estimates this parameter.

Summary

- We understand about Metamodeling.
- Credits, Jerry Banks book.

Modeling and Simulation

Simple Linear Regression

Overview

- **Here, we will understand about the “Simple Linear Regression”.**

Simple Linear Regression

- The relationship is between independent variable x and a dependent variable y.
- Linear relationship.
- Value of y for given x is:

$$E(Y \mid x) = \beta_0 + \beta_1 x$$

- β_0 intercept on y axis.
- β_1 is the slope.

Simple Linear Regression

- Y can be described as:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

- ε is random error with mean zero.
- This model is commonly called a simple linear regression model.

Simple Linear Regression

- n **pair of observations** $(y_1, x_1), (y_2, x_2) \ldots (y_n, x_n)$
- β_0 **and** β_1 **estimates the observation.**
- **Sum of squares of the deviation and regression line is minimized.**

Simple Linear Regression

- **The individual observations:**

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i,\ i = 1, 2, \ldots, n$$

- $\varepsilon_1, \varepsilon_2$ **... are uncorrelated random variables.**
- ε_i **is given by:**

$$\epsilon_i = Y_i - \beta_0 - \beta_1 x_i$$

- **It shows how** ε_i **is related to** x_i, Y_i **and** $E(Y_i|x_i)$**.**

Simple Linear Regression

- The sum of squares of the deviation is :

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_i)^2$$

- Where L is called least-squares function.
- Rewriting Y_i as:

$$Y_i = \beta_0' + \beta_1 (x_i - \bar{x}) + \epsilon_i$$

- Where:

$$\beta_0' = \beta_0 + \beta_1 \bar{x} \text{ and } \bar{x} = \sum_{i=1}^{n} x_i / n.$$

Simple Linear Regression

- It is often called as Transformed linear regression model.

$$L = \sum_{i=1}^{n} [Y_i - \beta_0' - \beta_1 (x_i - \bar{x})]^2$$

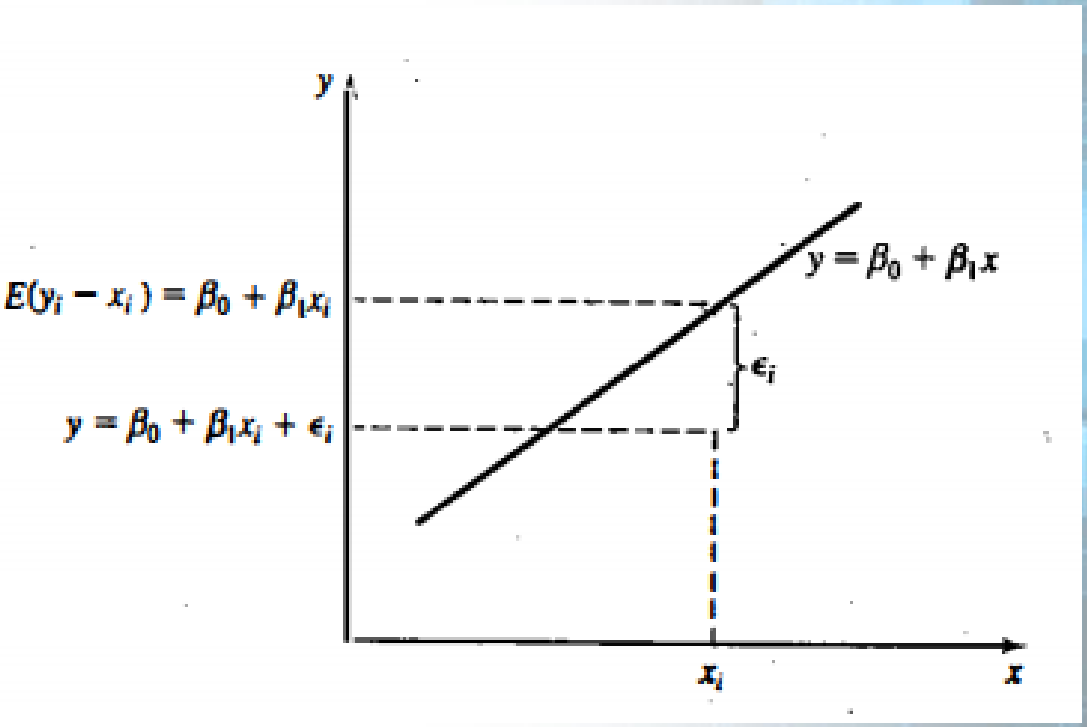
- Taking partial derivatives and setting each to zero.

$$n\hat{\beta}_0' = \sum_{i=1}^{n} Y_i$$

$$\hat{\beta}_1 \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} Y_i (x_i - \bar{x})$$

- These are called normal equations.

Simple Linear Regression



Relationship of ε_i [[**to** x]]$_i$,y_i and $E(\vdash Y_i \dashv \mid x_i)$.

Summary

- Here we understand about Simple Linear Regression.
- Credits: Uri Wilensky book

Modeling and Simulation

Testing for Significance of Regression

Overview

- **Here, we will understand about the “Testing for Significance of Regression”.**

Testing for Significance of Regression

- **Testing for the significance of regression provides another means for assessing the adequacy of the model.**
- **Error component ε_i is normally distributed.**

Testing for Significance of Regression

- **The adequacy of the model is checked by residual analysis.**
- **Developed by variance properties of and β_0 and β_1.**

Testing for Significance of Regression

- **Alternative hypothesis:**

$$H_0 : \beta_1 = 0$$
$$H_1 : \beta_1 \neq 0$$

- H_0 **indicates no relationship between x and Y.**
- **x is of little value in explaining the variability in Y.**

Testing for Significance of Regression

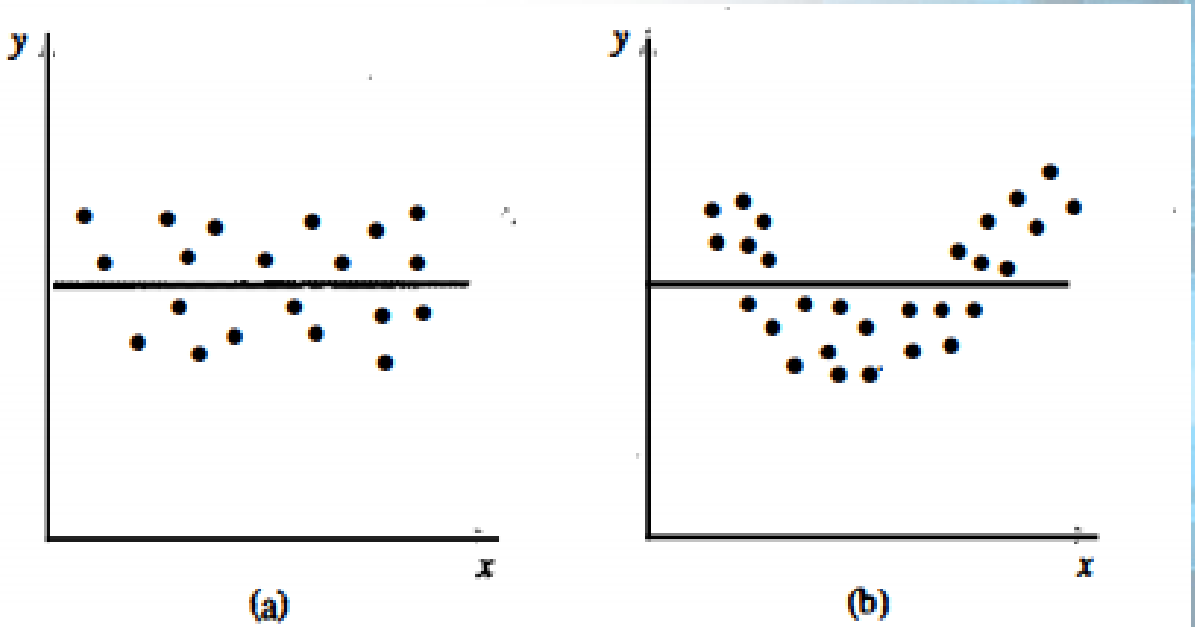


Figure 12.5 Failure to reject $H_0 : \beta_1 = 0$.

Illustration Best Estimator

Testing for Significance of Regression

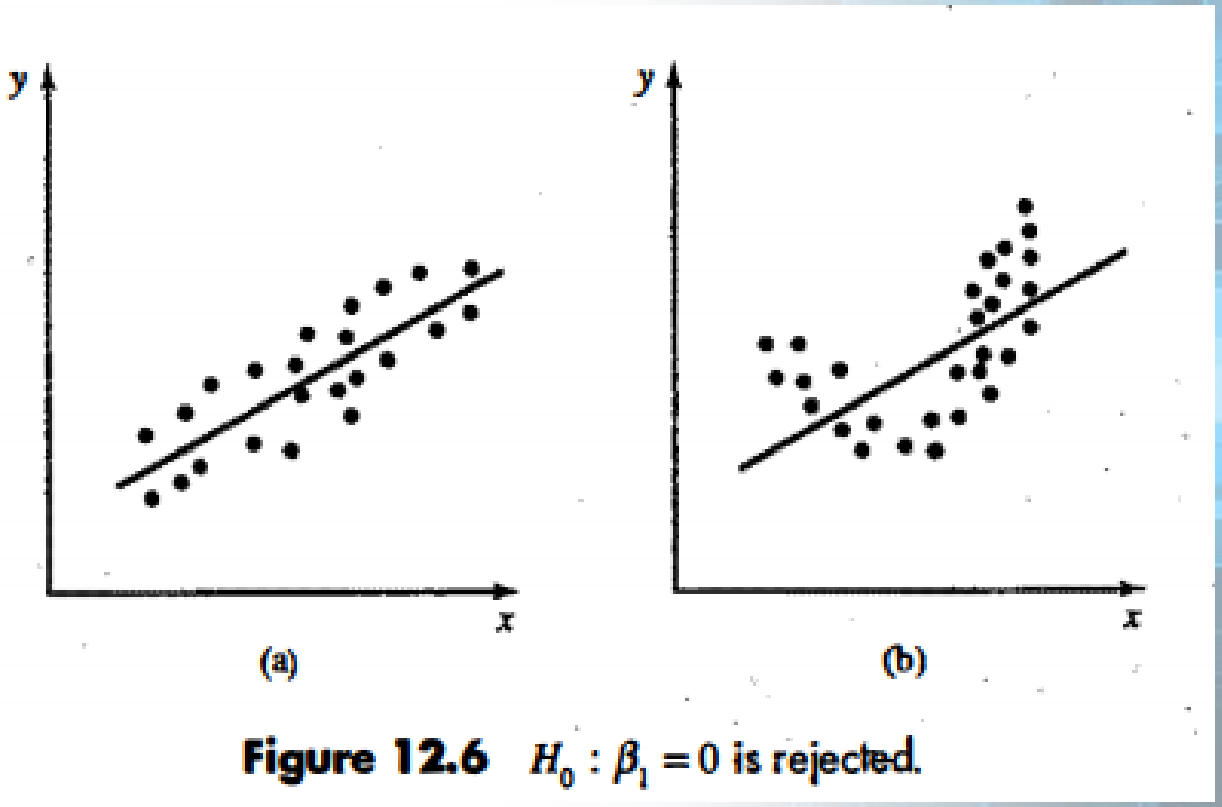


Figure 12.6 $H_0 : \beta_1 = 0$ is rejected.

Illustration Non-linear relationship

Testing for Significance of Regression

- **Testing for Significance of Regression**
- **Statistic for significance of regression:**

$$t_0 = \frac{\hat{\beta}_1}{\sqrt{MS_\varepsilon / S_{xx}}}$$

- MS_ε **is mean squared error given by:**

$$\mathrm{MS}_\varepsilon = \sum_{i=1}^{n} \frac{e_i^2}{n-2}$$

Testing for Significance of Regression

- **The direct method calculates $\sum_{i=1}^{n} e_i^2$.**

$$\sum_{i=1}^{n} e_i^2 = S_{yy} - \hat{\beta}_1 S_{xy}$$

- **Where S_{yy} is corrected sum of squares Y given by:**

$$S_{yy} = \sum_{i=1}^{n} Y_i^2 - \frac{\left(\sum_{i=1}^{n} Y_i\right)^2}{n}$$

Summary

- Here, we understood about the “Testing for Significance of Regression”.
- Credits: Jerry banks book.

Modeling and Simulation

What Does 'Optimization via Simulation' Mean?

Overview

- **Here, we will understand about the “Optimization via Simulation”.**

Optimization via Simulation

- Optimization is a key tool.
- Well developed algorithms for classes of problems.
- Linear Programming is Most famous.
- Much of optimization deals with performance of design.

Optimization via Simulation

- In discrete event simulation, result is a random variable.
- Let $x_1, x_2 \cdots x_n$ be $\boldsymbol{m}$ controllable design variables.
- Let $\boldsymbol{Y}(x_1, x_2 \cdots x_n)$ be observed simulation output performance.

Optimization via Simulation

- $x_1, x_2 \cdots x_n$ **might denote number of Automated Guided Vehicles.**
- $\boldsymbol{Y}(x_1, x_2 \cdots x_n)$ **could be total Material Handling System acquisition and operation cost.**

Optimization via Simulation

- $Y(x_1, x_2 \cdots x_n)$ **with respect to** $x_1, x_2 \cdots x_n$.
- **Y is random variable, so cannot optimize actual value of Y.**
- **Definition of optimization.**

maximize or minimize $E(Y(x_1, x_2, \ldots, x_m))$

Optimization via Simulation

- $\mathbf{E}(\boldsymbol{Y}(x_1, x_2 \cdots x_n))$ is long-run average cost of operating the MHS .
- x_1 AVGs, x_2 load per average, and routing algorithm x_3.
- Formulation of performance measure.

$$Y'(x_1, x_2, x_3) = \begin{cases} 1, & \text{if } Y(x_1, x_2, x_3) \le D \\ 0, & \text{otherwise} \end{cases}$$

Optimization via Simulation

- 1. Combine all of the performance measures into a single measure.
- The most common being cost.
- 2. Optimize with respect to one key performance measure.
- Evaluate with secondary performance measures.

Optimization via Simulation

- **3. Optimize with respect to one key performance measure.**
- **Consider those meeting certain constraints on performance.**
- **Expected cost and cycle time.**

Summary

- Here, we understood about Optimization via Simulation.
- Credits: Jerry banks book.

Modeling and Simulation

Difficulties in Optimization via Simulation

Overview

- **Here, We will understand “Why is Optimization via Simulation Difficult?”.**

Difficulties in Optimization via Simulation

- If design variables are numerous optimization can be difficult.
- A diverse collection of design variable types is known as the structure of performance function
- The performance can not be evaluated , but can be estimated.

Difficulties in Optimization via Simulation

- No conclusion with estimates.
- It can frustrate optimization algorithms that try to move in improving directions.
- It can be eliminated using many replications.
- Or using long runs.

Difficulties in Optimization via Simulation

- At each design point that the performance estimate has essentially no variance.
- It causes other designs to be explored.

Difficulties in Optimization via Simulation

- Standard sampling variability force optimizations are as following.
- Guarantee a prespecified probability of correct selection.
- Guarantee asymptotic convergence.

Difficulties in Optimization via Simulation

- **Optimal for deterministic counterpart.**
- **Robust heuristics.**

Difficulties in Optimization via Simulation

- **Robust heuristics are the most common algorithms.**
- **It is found in commercial optimization via simulation software.**

Summary

- Here, we understood about Bonferroni Approach to Selecting the Best.
- Credits, Jerry Banks book.

Modeling and Simulation

Using Robust Heuristics

Overview

- **Here, we will discuss about “Using Robust Heuristics”.**

Using Robust Heuristics

- It means a procedure that does not depend on strong problem structure.
- Can be applied on decision variables and sampling variability.
- Generic algorithms and tabu search are examples.

Using Robust Heuristics

- There are k possible solutions to the optimization via simulation problem.
- On each iteration a GA operates on a "population" of p solutions.

Using Robust Heuristics

- **Basic Genetic Algo**
- **Set the iteration counter j = 0.**
  $p(0) = \{x_1(0) \ldots x_p(0)\}$
- **Ruin simulation experiments.**
- **Select population p from P(j).**
- **Recombine Solutions via P(j + 1).**
- **Set j = j + 1 and repeat.**

Using Robust Heuristics

- The G.A can be terminated after some number of iterations.
- At termination X* with smallest Y(x) is choosen as best.

Using Robust Heuristics

- Basic Tabu Search
- Set the iteration counter j = 0. select initial x* in X
- Find solution x’ which minimizes Y(x).
- If Y(x‘) < Y (x*) then x* = x’.
- Update the list of Tabu and repeat from step 2.

Using Robust Heuristics

- Tabu search can be terminated when some number of iterations has passed without changing x*.
- T.S is fundamentally a discrete-decision-variable optimizator.

Summary

- Here, we have understood concepts about Robust Heuristics
- Credits: Jerry banks book.

Modeling and Simulation

Random Search: An Illustration

Overview

- **Here, we will understand About “An Illustration: Random Search”.**

Random Search: An Illustration

- Random search provides guaranteed asymptotic converges in certain conditions.
- Convergence can be slow and memory consumptive.
- random search is not a Robust Heuristic.

Random Search: An Illustration

- Random search algorithm requires finite number of possible designs.
- Rule out problem with continuous decision variables.
- Algorithms designed for continuous variable problem is there.

Random Search: An Illustration

- On each iteration, we compare a current solution to a randomly chosen competitor.
- If competitor is better it becomes current good solution.

Random Search: An Illustration

- **Step 1.**
- **Initialize random variables $\mathbf{C}(i) = \mathbf{0}$.**
- **Initial solution i^0 and set $\mathbf{C}(i^0) = 1$.**
- **$\mathbf{C}(i)$ count number of times i is visited.**

Random Search: An Illustration

- Step 2.
- Chose another solution except i .
- Such as each solution has equal chance of being selected.

Random Search: An Illustration

- **Step 3.**
- **Run two simultaneous experiments.**
- **Use i^0 and i to obtain $Y(i^0)$ and $Y(i)$**
- **Set $i^0 = i$**

Random Search: An Illustration

- **Step 4.**
- **Set** $C(\boldsymbol{i}^0) = C(\boldsymbol{i}^\circ) + 1$
- **If not done go to step 2.**
- **If done, Then select** $x\boldsymbol{i}^*$ **such that** $\boldsymbol{C}(\boldsymbol{i}^*)$ **is largest count.**

Summary

- We understood about Random search.
- Credits, Jerry Banks book.

Modeling and Simulation

Introduction to performance evaluation

Overview

- **Here, we will see the introduction to performance evaluation.**

Introduction to performance evaluation

- Understanding all elements of the computer system.
- All aspects are important while understanding the issues.
- Realizing the requirements.

Introduction to performance evaluation

- Computer system must function correctly.
- Performing the intended function efficiently.

Introduction to performance evaluation

- Describing the computer system's design.
- Construction.
- Fielding.
- Life-cycle maintenance.

Summary

- We have learnt about the introduction to the performance evaluation.
- Credits: Paul Fortier book.

Modeling and Simulation

Events

Overview

- **Here, we will understand about the “Events”.**

Events

- Time is an important measure.
- Useful if we can measure under computer evaluation.
- An event describes an entity of interest in our system.
- E.g. Clock cycle.

Events

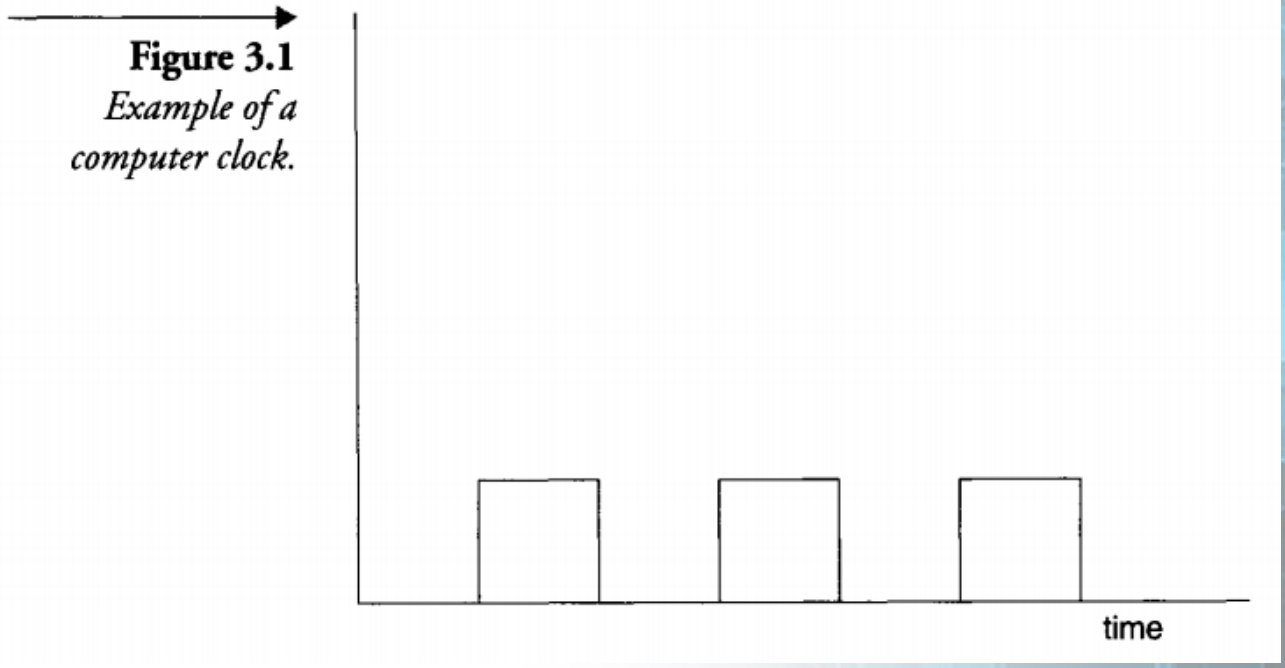


Figure 3.1
Example of a computer clock.

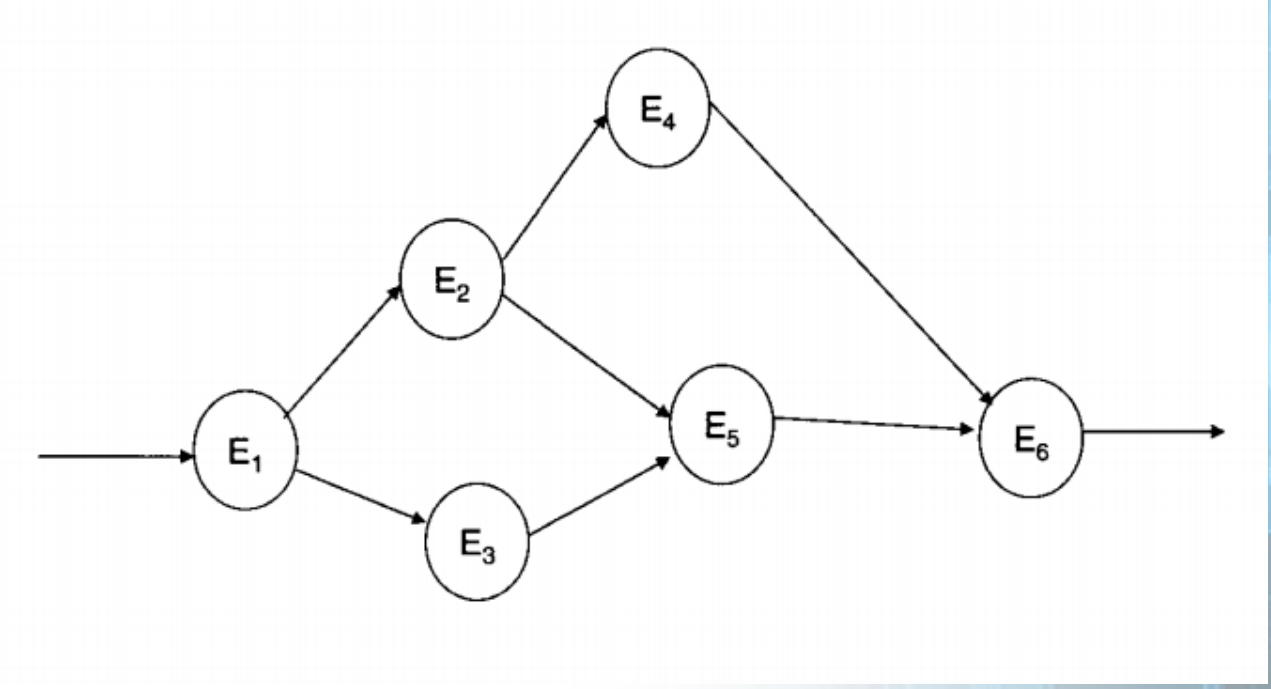
Events

- All events must be controlled.
- Each action is designed to synchronize with other actions.
- Small actions makes larger events.

Events

- System analyst must have understanding of the events.
- And the relationship among different events.

Events



Summary

- Here we understand about Events.
- Credits: Uri Wilensky book

Modeling and Simulation

Measurements (sampling)

Overview

- **Here, we will learn about the measurements (sampling).**

Measurements (sampling)

- Events represent all of the real actions that occur in the computer system.
- Must know all the possible conditions.

Measurements (sampling)

- State is an important concept in any computer system.

$$S = \{E_1(\text{value}), E_2(\text{value}), E_3(\text{value}), \ldots, E_n(\text{value})\}$$

- Each event must have valid components.

Measurements (sampling)

- Primary types of measures are:
- Count a number of items over a given time period.
- Measures all state variables.
- Measures the fraction of time the system is within a state.
- Valid state must reach.

Measurements (sampling)

- Many ways are there to measure system events and decision depends on many factors.
- In hardware monitoring system analyst has ability to add instrumentation.
- Can also use integral test hardware.

Measurements (sampling)

- **The hardware monitoring be designed as an integral component of the system.**
- **Must determine and define all aspects.**

Measurements (sampling)

- Software monitoring requires designers to system hardware elements.
- As well as low level software elements.
- Need to have access to low level operating system calls.

Measurements (sampling)

- **Hybrid monitoring uses the concepts of both software and hardware monitoring.**
- **Synchronization is required.**

Summary

- We have learnt about the Measurements (sampling).
- Credits: Paul Fortier book.

Modeling and Simulation

Intervals

Overview

- **Here, we will learn about the intervals.**

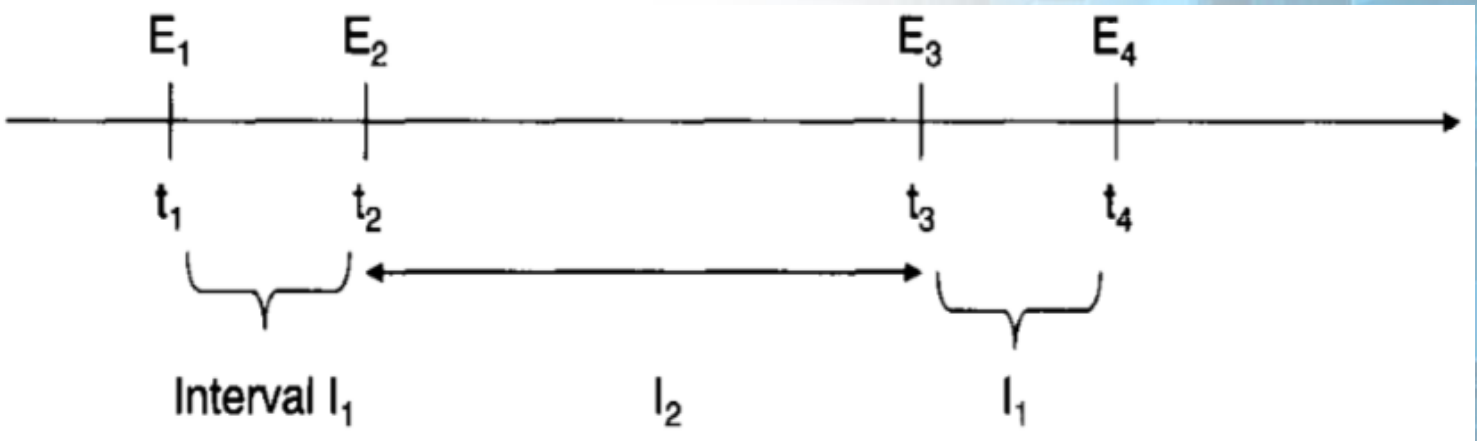
Intervals

- An interval represents a period of time.
- System environment is system clocks.
- Instruction execution cycle.

Intervals

- Intervals are same if they represent the same sequence of events.
- Separate sequence are represented by two intervals then they are unrelated.

Intervals



Summary

- We have learnt about the Intervals.
- Credits: Paul Fortier book.

Modeling and Simulation

Response

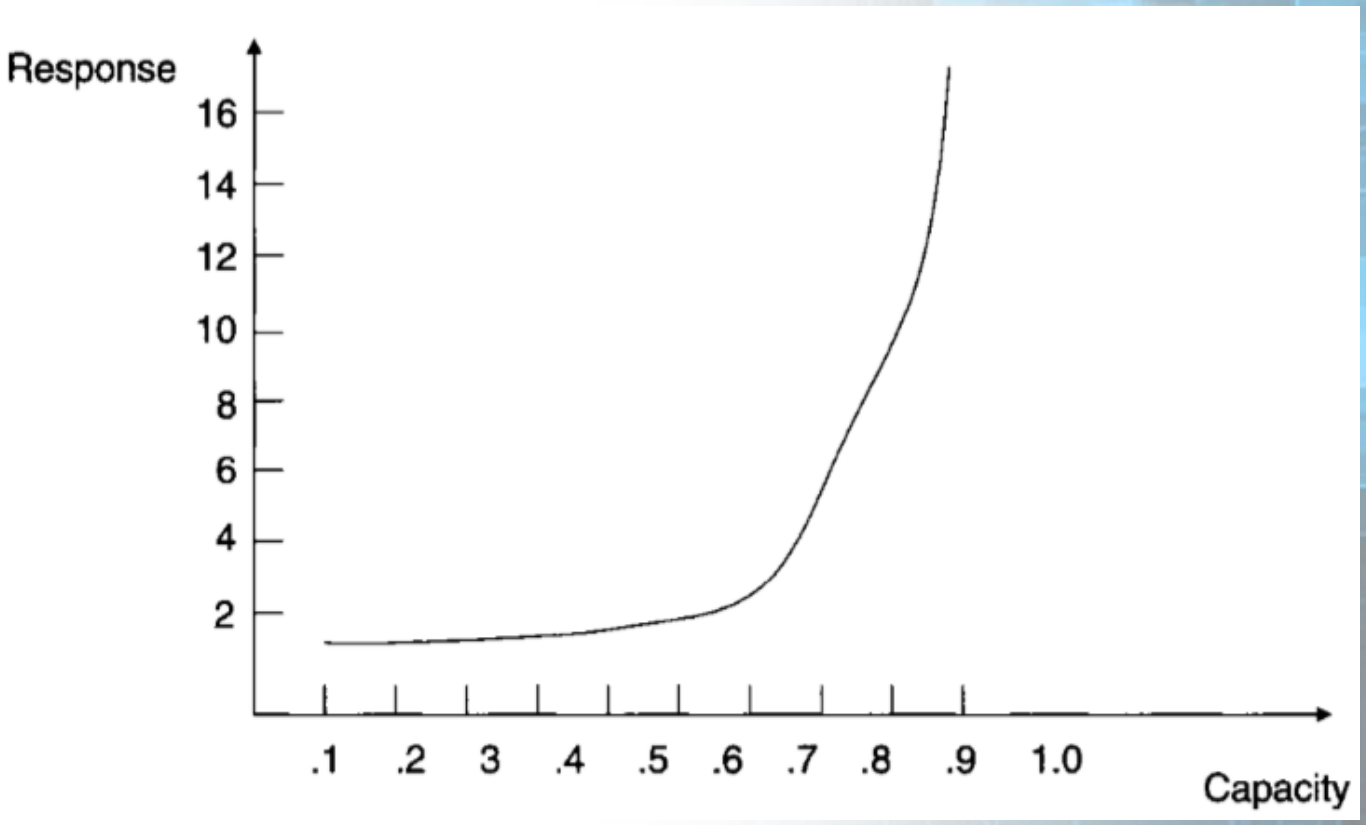
Overview

- **Here, we will learn about the response.**

Response

- Represent a measure of period of time a user or application to wait before task assignment.
- Typical measure may pit the response time.

Response



Response

- Load increases above specific capacity the response increases.
- Study the measurable sequences against system load.
- How these loads impact each other.

Summary

- We have learnt about the response.
- Credits: Paul Fortier book.

Modeling and Simulation

Independence

Overview

- **Here, we will learn about the Independence.**

Independence

- **Occurrence if one dose not influence the outcome of other.**

Independence

- Important concept while evaluating systems.
- Two programs that cannot run concurrently must be consider independent.

Independence

- Running on the same operating system and hardware.
- Cannot interfere each other then they are separate and unrelated.

Independence

- **Important in modeling to define all elements and their relationship.**

Summary

- We have learnt about the Independence.
- Credits: Paul Fortier book.

Modeling and Simulation

Randomness

Overview

- **Here, we will learn about randomness.**

Randomness

- Randomness is a property of an event and its reoccurrence.
- Have no pattern.
- Realization of the property of independence.

Randomness

- Its a mathematical concept.
- External sources can be viewed as random events in systems.

Randomness

- **Important to analyze systems using mathematical concepts.**

Summary

- We have learnt about the randomness.
- Credits: Paul Fortier book.

Modeling and Simulation

Workloads

Overview

- **Here, we will learn about the workloads.**

Workloads

- Simply we can say load.
- Event or event sequences presented to system to model.
- Represent the number of events offered to system for execution.

Workloads

- Arranging the requests and perform them by the system.
- The duration could be endless.
- The load is re-entered after prescribed period of time.

Workloads

- **The database community has developed a set of transactional workloads.**

Summary

- We have learnt about the workloads.
- Credits: Paul Fortier book.

Modeling and Simulation

Problems encountered in model development and use

Overview

- **Here, we will understand about the problems encountered in model development and use.**

Problems encountered in model development and use

- Must start with having concept that what we are doing.
- Determining what to do.
- Two main classes performance assessment.
- First take an existing system and design some experiments.

Problems encountered in model development and use

- Second is either analytical modeling or simultaneous.
- Analyst uses following measures:
- Responsiveness and use levels.
- Mission-ability and dependability.
- Productivity and predictability.

Problems encountered in model development and use

- Analyst must avoid the following mistakes:
- Having no goals.
- Setting biased goals.
- Unsystematic approach to development.
- Choosing of incorrect performance metrics.
- Choosing non-stressful workload.

Problems encountered in model development and use

- **To solve these problems the analyst must ask some questions to himself/herself.**

Summary

- We have learnt about the problems encountered in model development and use.
- Credits: Paul Fortier book.

Modeling and Simulation

A Case Study

Overview

- **Here, we will understand about how to conduct a case study**

A Case Study

- First Step: Define the System that we wish to study
- E.g. Remote pipes vs. Remote Procedures

A Case Study

- **Focus on Services – small and large data transfers**

A Case Study

- Select Metrics
- Rate of access
- Time for Performance
- Resource requirements per service

A Case Study

- Evaluate the developed model and start over again
- Trial and error

A Case Study

- Conducting our own: Modeling a ? ? ?
- A Bus system
- A railway terminal
- A hospital
- A human cell
- Human brain
- Ecosystem

Summary

- We have learnt about conducting a case study.
- Credits: Paul Fortier book.

Modeling and Simulation

Introduction to Software Reliability

Overview

- **Here, we will understand about software reliability**

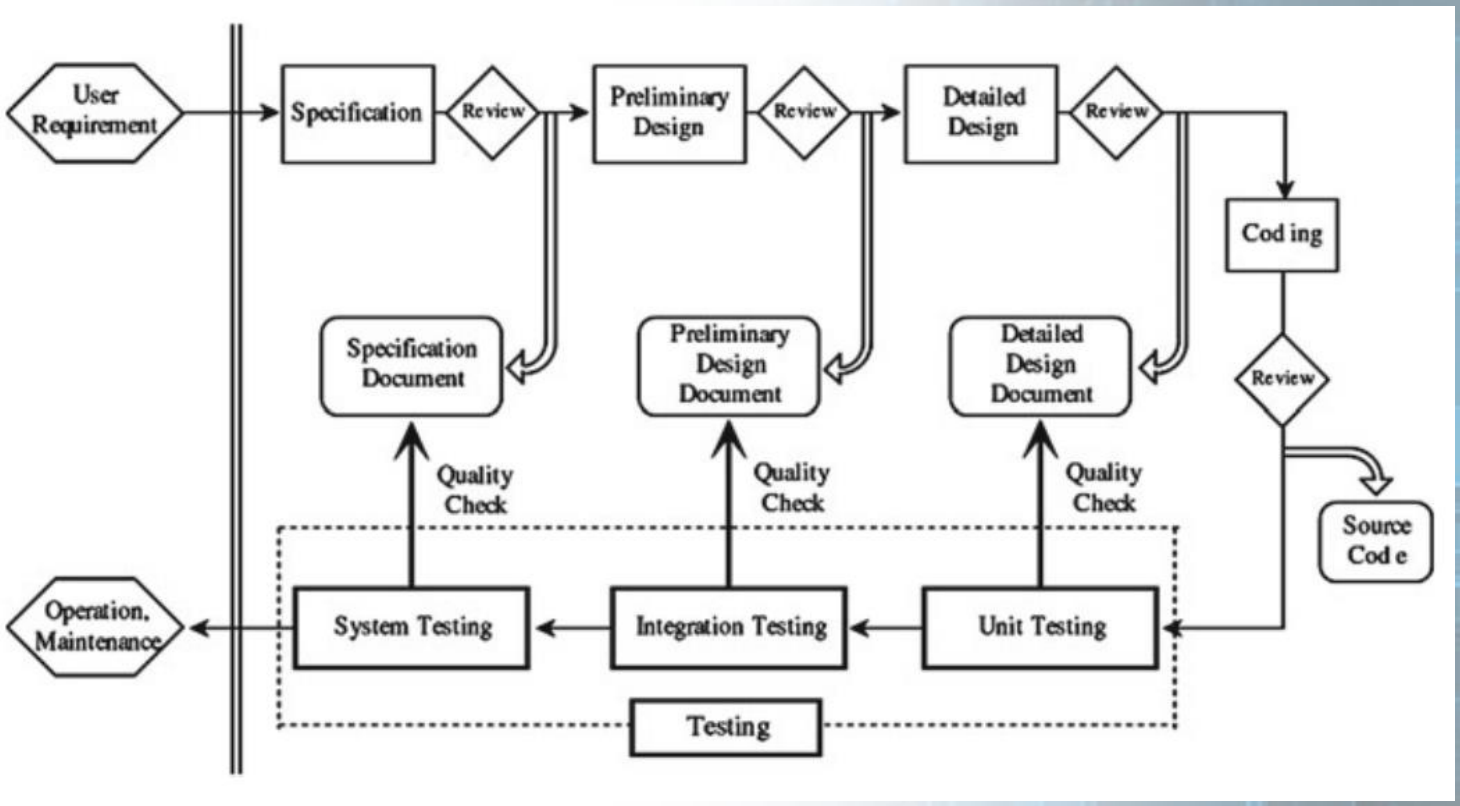
Introduction to Software Reliability

- Software fails on a regular basis
- This is a serious issue
- Especially in case of critical systems – nuclear reactors, patient care etc.

Introduction to Software Reliability

- Problems with X-Ray machine
- Problems with SCUD Missiles
- Problems with rocket launch
- Etc.

Introduction to Software Reliability



Summary

- We have learnt about software reliability.
- Credits: Yamada book.

Modeling and Simulation

Definitions and Software Reliability Model

Overview

- **Here, we will understand about definitions about reliability**

Definitions and Software Reliability Model

- **Generally, a software failure caused by software faults latent in the system cannot occur except on a specific occasion when a set of specific data is put into the system under a specific condition, i.e. the program path including software faults is executed.**

Definitions and Software Reliability Model

- **A software system is a product which consists of the programs and documents produced through a software development process**

Definitions and Software Reliability Model

- **Software failure is defined as an unacceptable departure of program operation from the program requirements.**
- **The cause of software failure is called a software fault.**

Definitions and Software Reliability Model

- Then, software fault is defined as a defect in the program which causes a software failure.
- The software fault is usually called a software bug.

Definitions and Software Reliability Model

- **Software error is defined as human action that results in the software system containing a software fault**

Definitions and Software Reliability Model

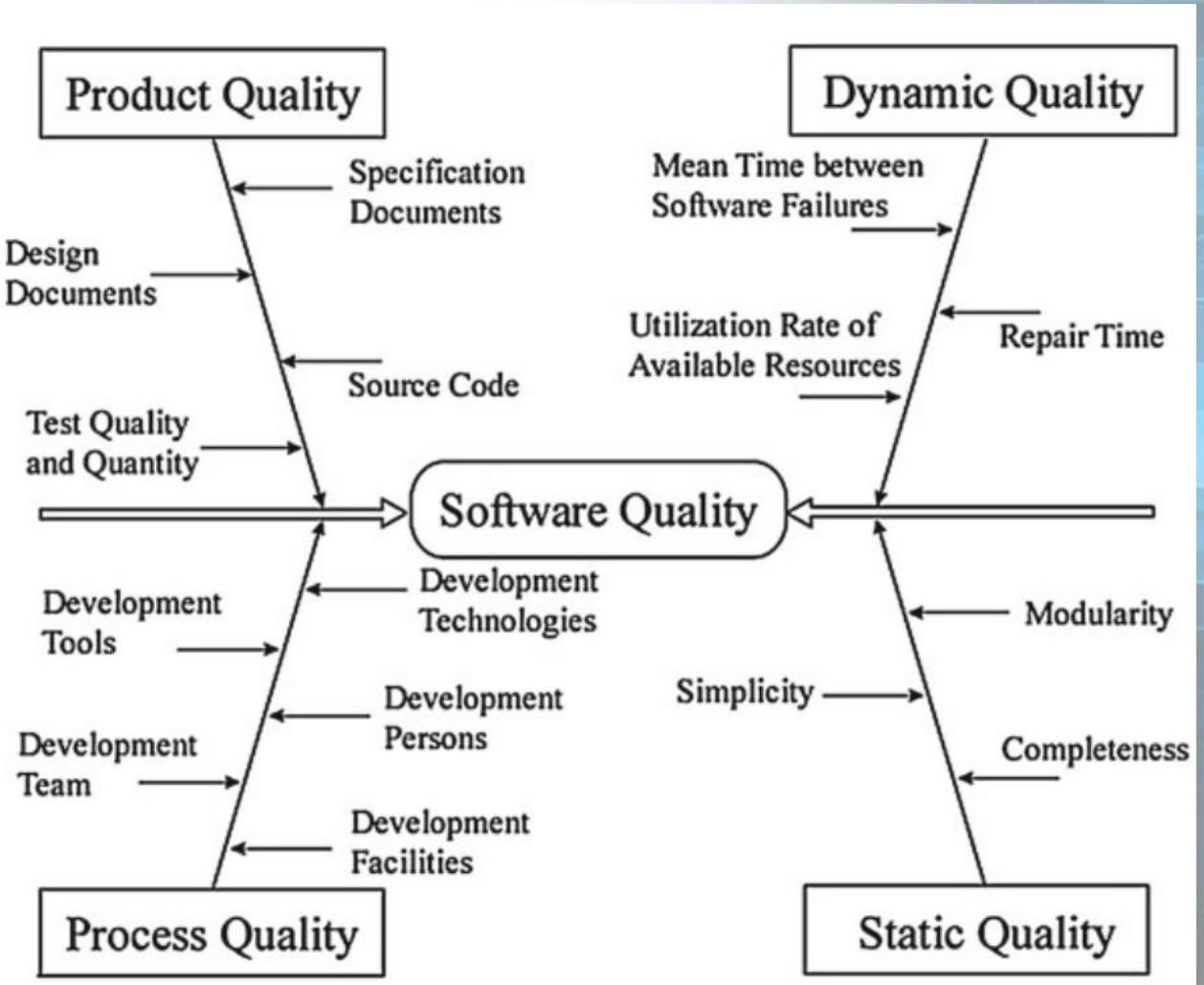
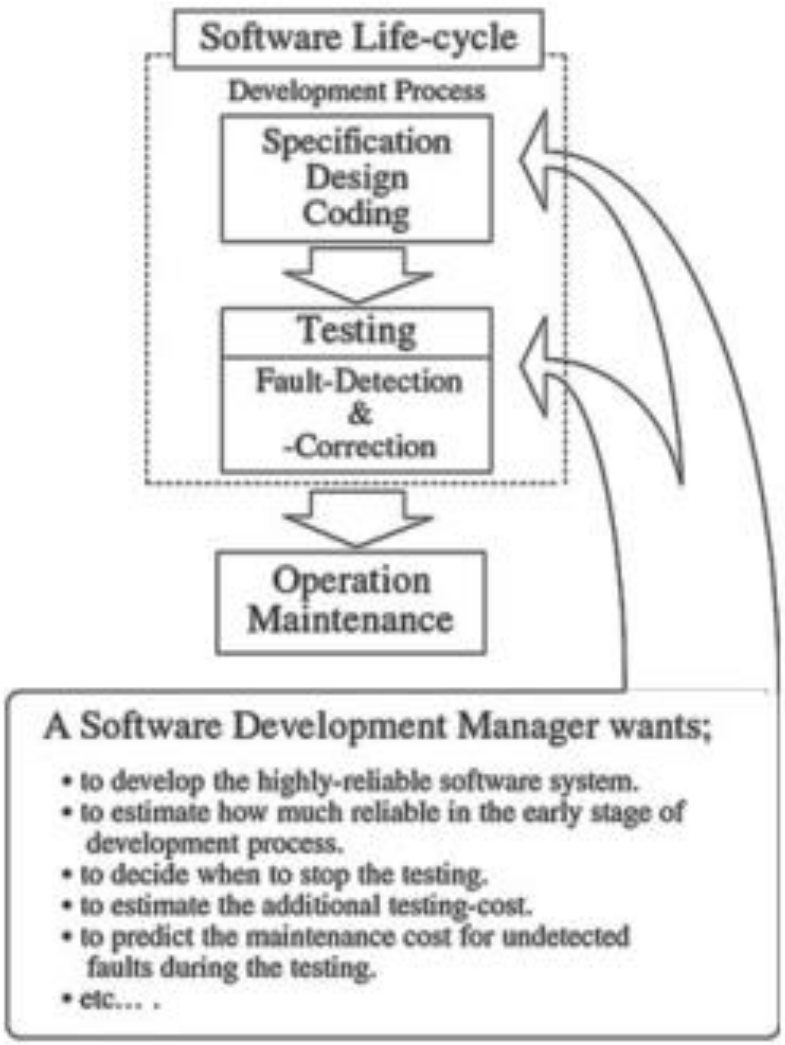


Fig. 1.2 Elements of software quality based on a cause-and-effect diagram

Definitions and Software Reliability Model

Fig. 1.3 Aim of software quality/reliability measurement and assessment



Definitions and Software Reliability Model

Terminologies

- *Software Failure*
  - — an unacceptable departure of program operation from the program requirements
- *Software Fault*
  - — a defect in the program which causes a software failure (*software bug*)
- *Software Reliability*
  - — the attribute that a software system will perform without causing software failures over a given time period under specified conditions
  - — measured by its probability

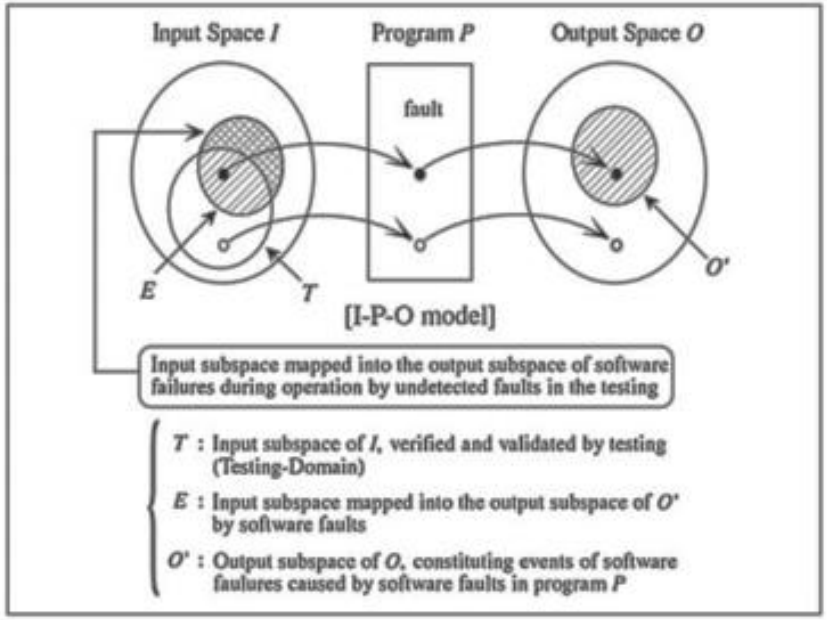


Fig. 1.4 An input-program-output model for software behavior

Definitions and Software Reliability Model

Software Reliability	Hardware Reliability
(1) Software failures can be due to no wearout phenomenon.	(1) Hardware failures can be due to wear.
(2) Software reliability is, inherently, determined during the earlier phase of the development process, i.e. specification analysis and design phases.	(2) Hardware reliability is affected by deficiencies injected during all phases of the development, operation, and maintenance.
(3) Software reliability can not be improved by redundancy with identical versions.	(3) Hardware reliability can be improved by redundancy with identical units.
(4) A verification method of software reliability has not been established.	(4) A testing method of hardware reliability is established and standardized.
(5) A maintenance technology is not established since the market of software products is rather recent.	(5) A maintenance technology is advanced since the market of hardware products is established and the user environment is seized.

Fig. 1.5 Comparison between the characteristics of software reliability and hardware reliability

Summary

- We have learnt about definitions related to reliability.
- Credits: Yamada book.

Modeling and Simulation

Software Reliability Growth Modeling

Overview

- **Here, we will understand about software reliability growth modeling**

Software Reliability Growth Modeling

- **Generally, a mathematical model based on stochastic and statistical theories is useful to describe the software fault-detection phenomena or the software failure-occurrence phenomena and estimate the software reliability quantitatively.**

Software Reliability Growth Modeling

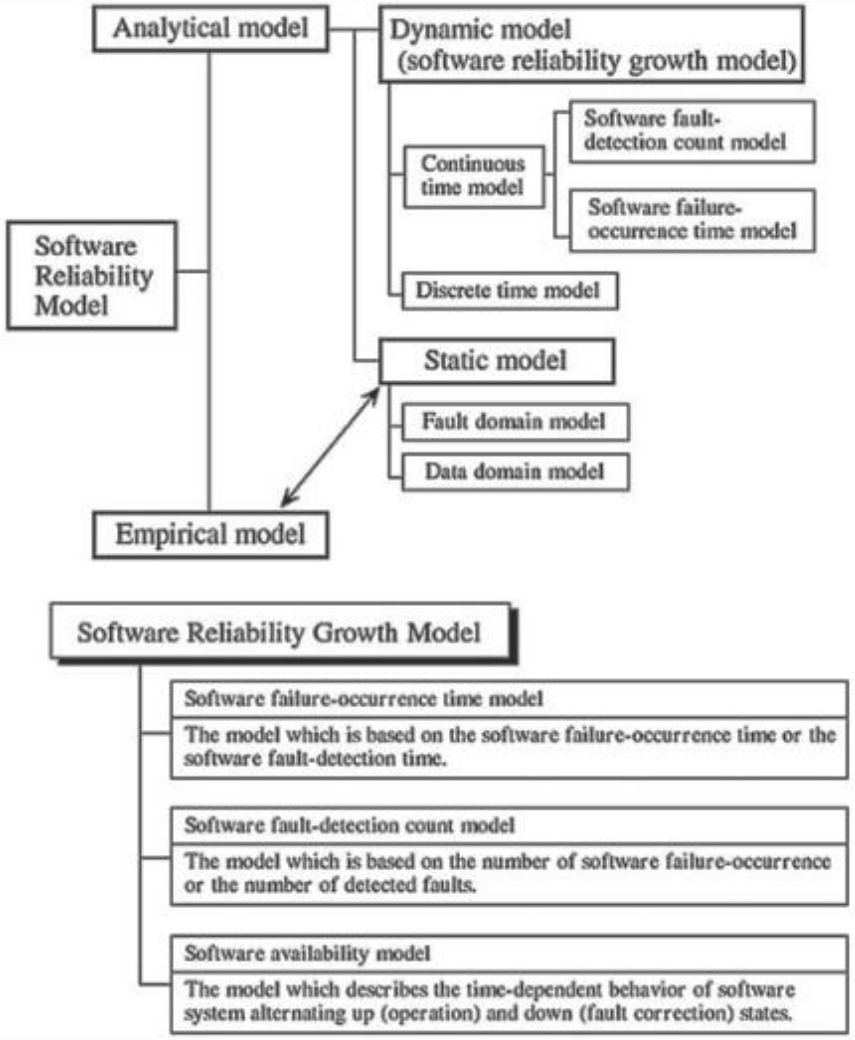


Fig. 1.6 Hierarchical classification of software reliability models

Software Reliability Growth Modeling

$N(t)$	the cumulative number of software faults (or the cumulative number of observed software failures) detected up to time t,
S_i	the ith software-failure occurrence time ($i = 1, 2, \ldots; S_0 = 0$),
X_i	the time-interval between $(i-1)$-st and ith software failures ($i = 1, 2, \ldots; X_0 = 0$).

Software Reliability Growth Modeling

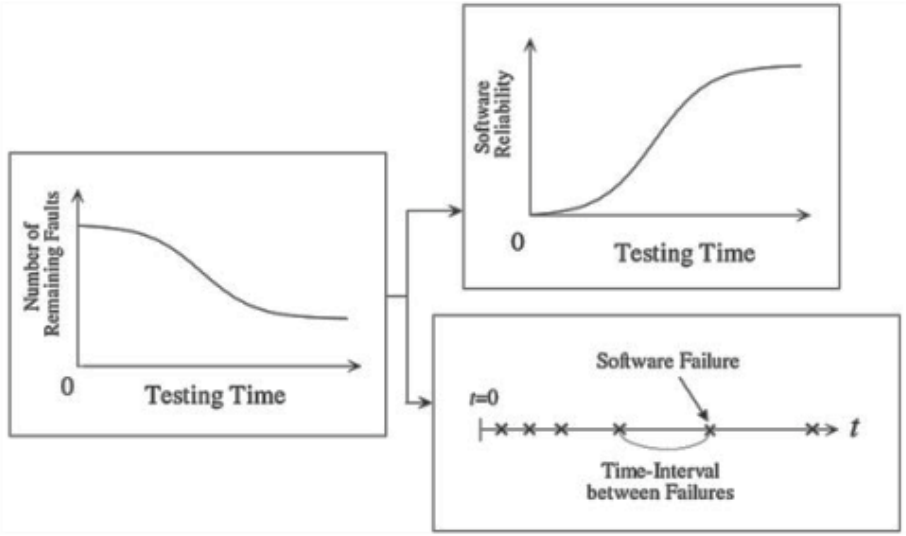


Fig. 1.7 Software reliability growth

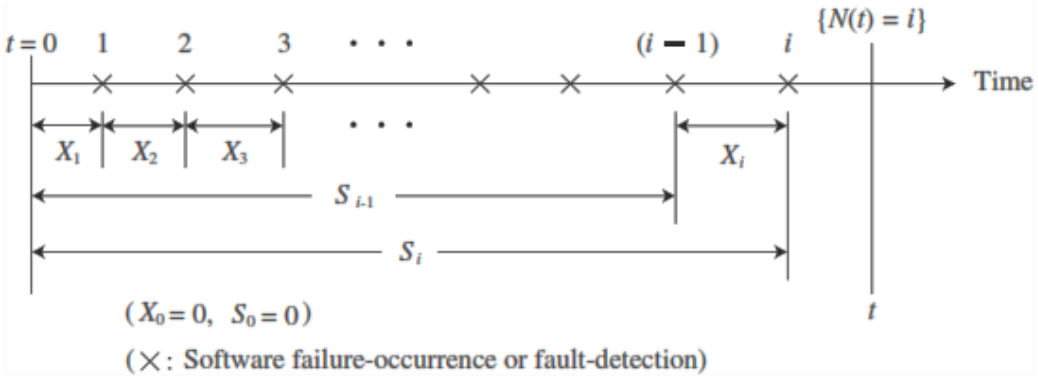


Fig. 1.8 The stochastic quantities related to a software fault-detection phenomenon or a software failure-occurrence phenomenon

Summary

- We have learnt about software reliability growth modeling.
- Credits: Yamada book.

Modeling and Simulation

Imperfect Debugging Modeling

Overview

- **Here, we will understand about imperfect debugging modeling**

Imperfect Debugging Modeling

- **Most software reliability growth models proposed so far are based on the assumption of perfect debugging**

Imperfect Debugging Modeling

- i.e. All faults detected during the testing and operation phases are corrected and removed perfectly.
- However, debugging actions in real testing and operation environment are not always performed perfectly.

Imperfect Debugging Modeling

- For example,
- Typing errors invalidate the fault-correction activity or fault-removal is not carried out precisely due to incorrect analysis of test results

Imperfect Debugging Modeling

- As a result it is preferable to develop an incorrect debugging model.
- This will result in a better estimation of reliability

Summary

- We have learnt about imperfect debugging modeling.
- Credits: Yamada book.

Modeling and Simulation

Imperfect Debugging Model with Perfect Correction Rate

Overview

- **Here, we will understand about imperfect debugging model with perfect correction rate**

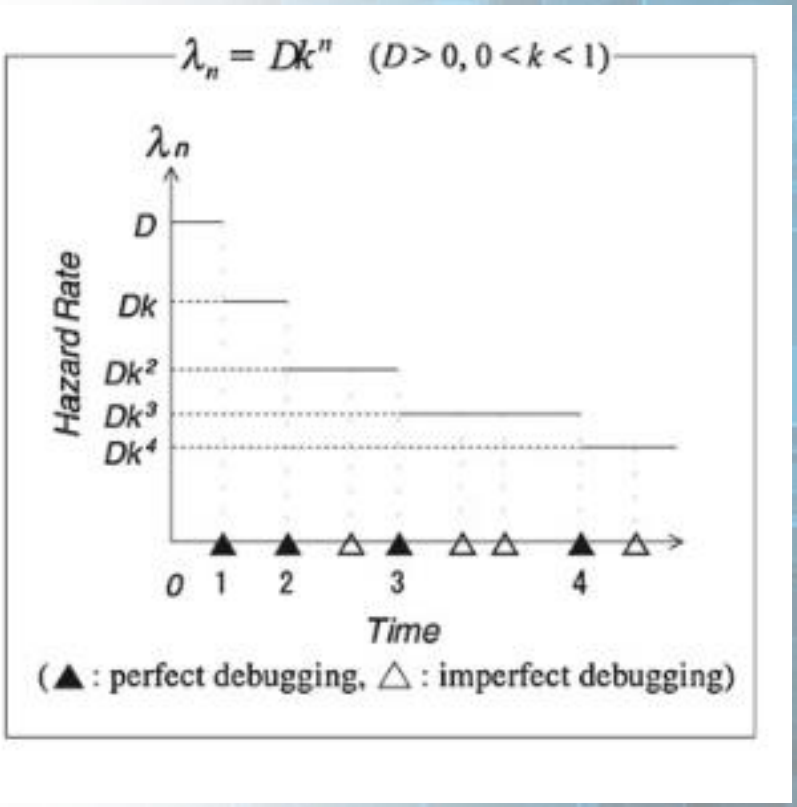
Imperfect Debugging Model with Perfect Correction

- **To model an imperfect debugging environment, the following assumptions are made:**

1. Each fault which causes a software failure is corrected perfectly with probability $p(0 \leq p \leq 1)$. It is not corrected with probability $q(= 1 - p)$. We call p the perfect debugging rate or the perfect correction rate.
2. The hazard rate is given by (1.5) and decreases geometrically each time a detected fault is corrected (see Fig. 1.14).
3. The probability that two or more software failures occur simultaneously is negligible.
4. No new faults are introduced during the debugging. At most one fault is removed when it is corrected, and the correction time is not considered.

Imperfect Debugging Model with Perfect Correction

Fig. 1.14 Behavior of hazard rate



Imperfect Debugging Model with Perfect Correction

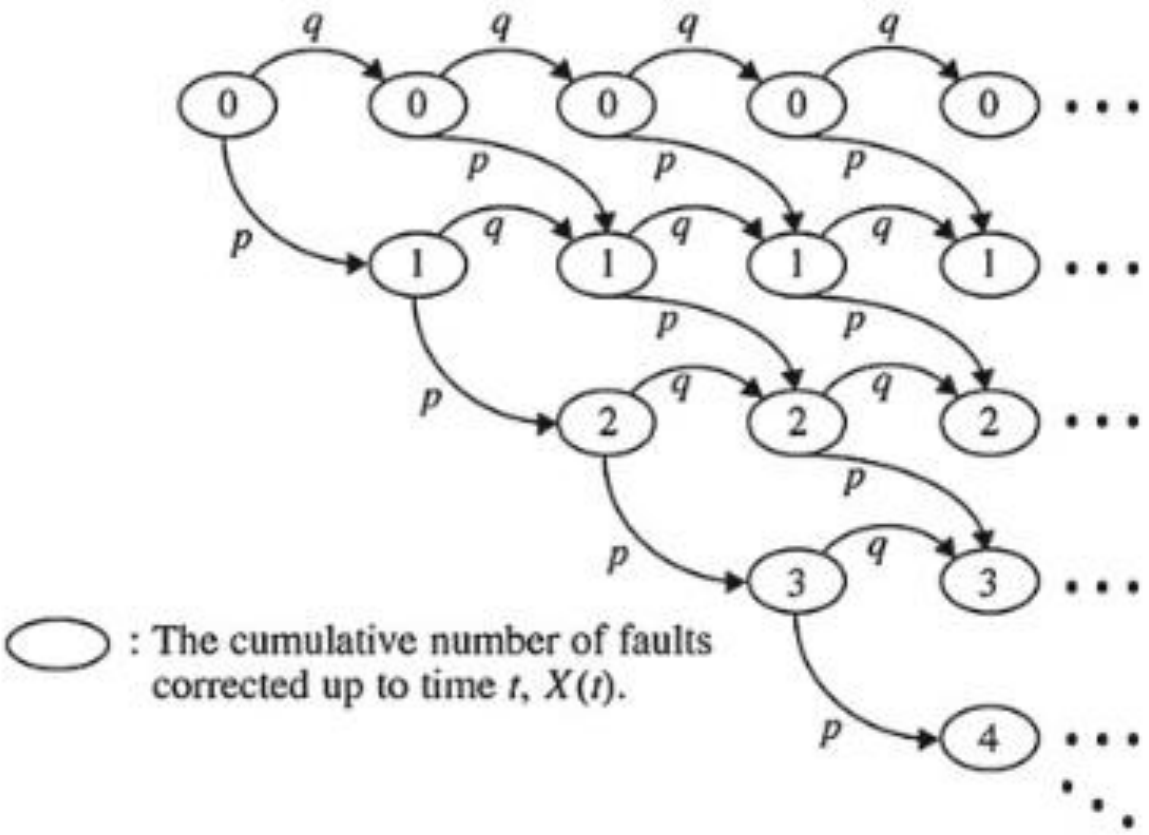


Fig. 1.15 A diagrammatic representation of transitions between states of $X(t)$

Imperfect Debugging Model with Perfect Correction

- Let X (t) be a random variable representing the cumulative number of faults corrected up to the testing time t. Then, X (t) forms a Markov process.
- That is, from assumption 1, when i faults have been corrected by arbitrary testing time t,

Imperfect Debugging Model with Perfect Correction

$$X(t) = \begin{cases} i, & \text{with probability } q, \\ i+1, & \text{with probability } p, \end{cases} \tag{1.14}$$

Summary

- We have learnt about imperfect debugging modeling with correction.
- Credits: Yamada book.

Modeling and Simulation

Imperfect Debugging with Introduced Faults

Overview

- **Here, we will understand about imperfect debugging modeling with introduced faults**

Imperfect Debugging with Introduced Faults

- Besides the imperfect debugging factor above in fault-correction activities, we consider the possibility of introducing new faults in the debugging process.
- It is assumed that two kinds of software failures exist in the dynami environment

Imperfect Debugging with Introduced Faults

- **(F1) software failures caused by faults originally latent in the software system prior to the testing (which are called inherent faults)**
- **(F2) software failures caused by faults introduced during the software operation owing to imperfect debugging.**

Imperfect Debugging with Introduced Faults

- In addition, it is assumed that one software failure is caused by one fault
- And that it is impossible to discriminate whether the fault that caused the software failure that has occurred is F1 or F2.

Imperfect Debugging with Introduced Faults

- **As to the software failure-occurrence rate due to F1, the inherent faults are detected with the progress of the operation time.**

Imperfect Debugging with Introduced Faults

- In order to consider two kinds of time dependencies on the decreases of F1, let
- ai(t) = (1,2) denote software failure occurrence rate for F1
- For F2, it is constant (Lambda > 0)

Summary

- We have learnt about imperfect debugging modeling with introduced faults.
- Credits: Yamada book.

Modeling and Simulation

Software Availability Modeling

Overview

- **Here, we will understand about software availability modeling**

Software Availability Modeling

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Software Availability Modeling

- That is, it is very important to measure and assess software availability,
- It is defined as the probability that the software system is performing successfully, according to the specification, at a specified time point

Software Availability Modeling

- **The actual operational environment needs to be more clearly reflected in software availability modeling, since software availability is a customer-oriented metrics.**

Software Availability Modeling

- Software Reliability :

> the attribute that software systems can continue to perform and do not cause any software failures *for a given time period*, under a specified environment.

■ *software failure*

[an unacceptable departure from program operation caused by a software fault remaining in the software system.]

- Software Availability :

> the attribute that software systems are available and performing *at a given time point*, under a specified environment.

Software Availability Modeling

- Software fails on a regular basis
- This is a serious issue
- Especially in case of critical systems – nuclear reactors, patient care etc.

Summary

- We have learnt about conducting a case study.
- Credits: Paul Fortier book.

Modeling and Simulation

Model Description

Overview

- **Here, we will understand about model description for software availability modeling**

Model Description

- **Assumptions**
- **1. The software system is unavailable and starts to be restored as soon as a software failure occurs, and the system cannot operate until the restoration action is complete**

Model Description

- 2. The restoration action implies debugging activity, which is performed perfectly with probability a
- (0 < a ≤ 1) and imperfectly with probability b(= 1 – a).
- a is the perfect debugging rate

Model Description

- **One fault is corrected and removed from the software system when the debugging activity is perfect.**

Model Description

- **3. When n faults have been corrected, the time to the next software failure- occurrence and the restoration time follow exponential distributions**

Model Description

4. The probability that two or more software failures will occur simultaneously is negligible

Summary

- We have learnt about model description.
- Credits: Yamada book.

Modeling and Simulation

Software Availability Measures

Overview

- **Here, we will understand about software availability measures**

Model Description

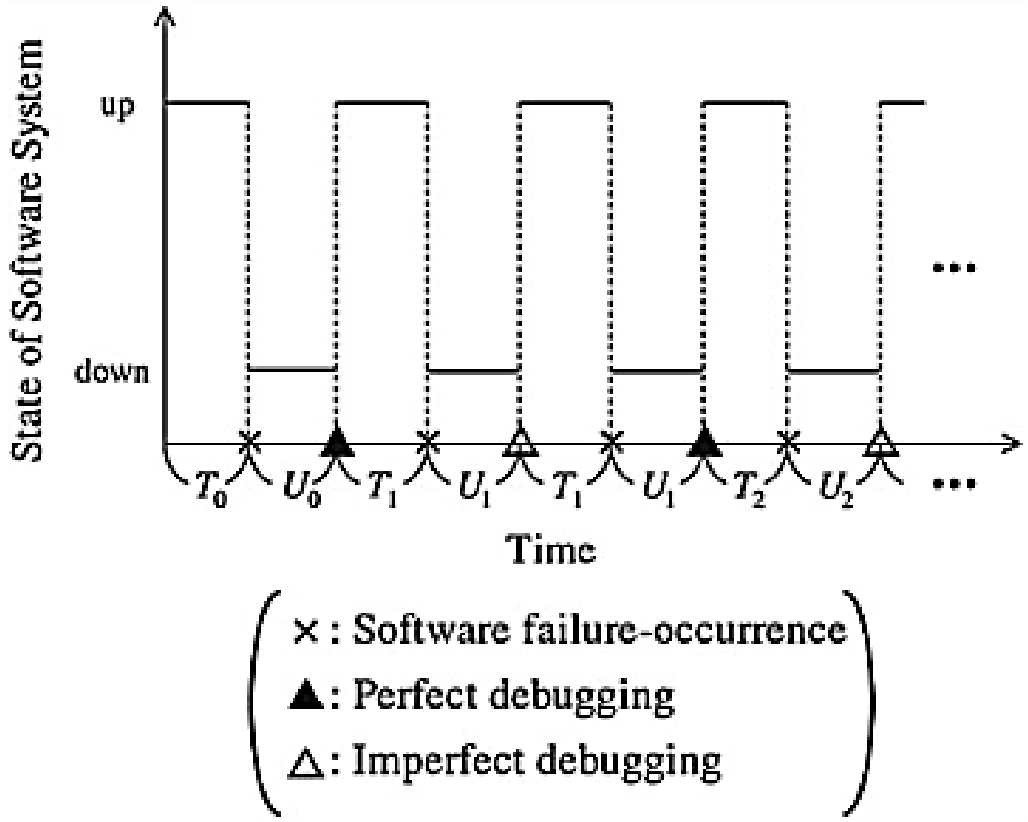


Fig. 1.17 Sample behavior of the software system alternating between up and down states

Software Availability Measures

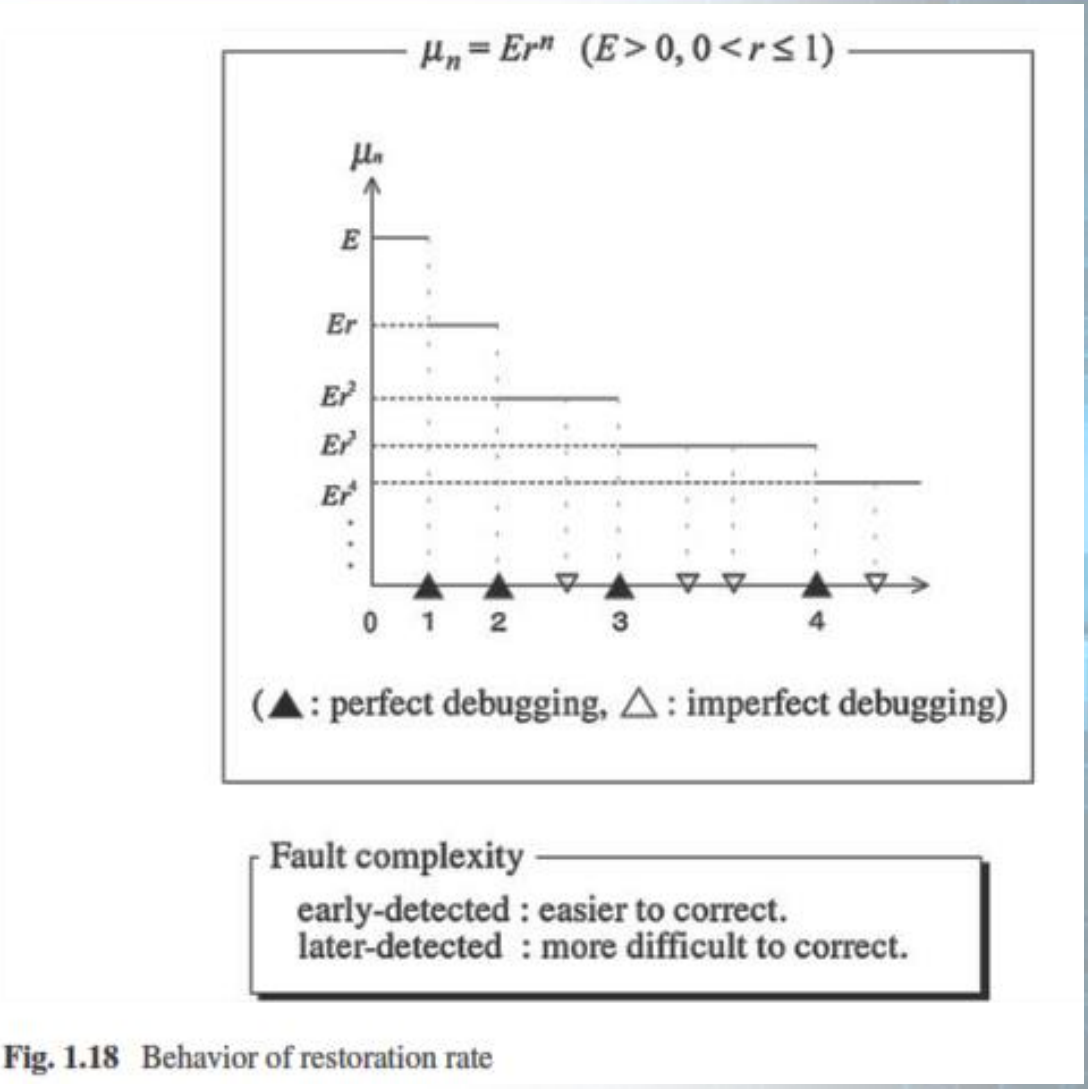
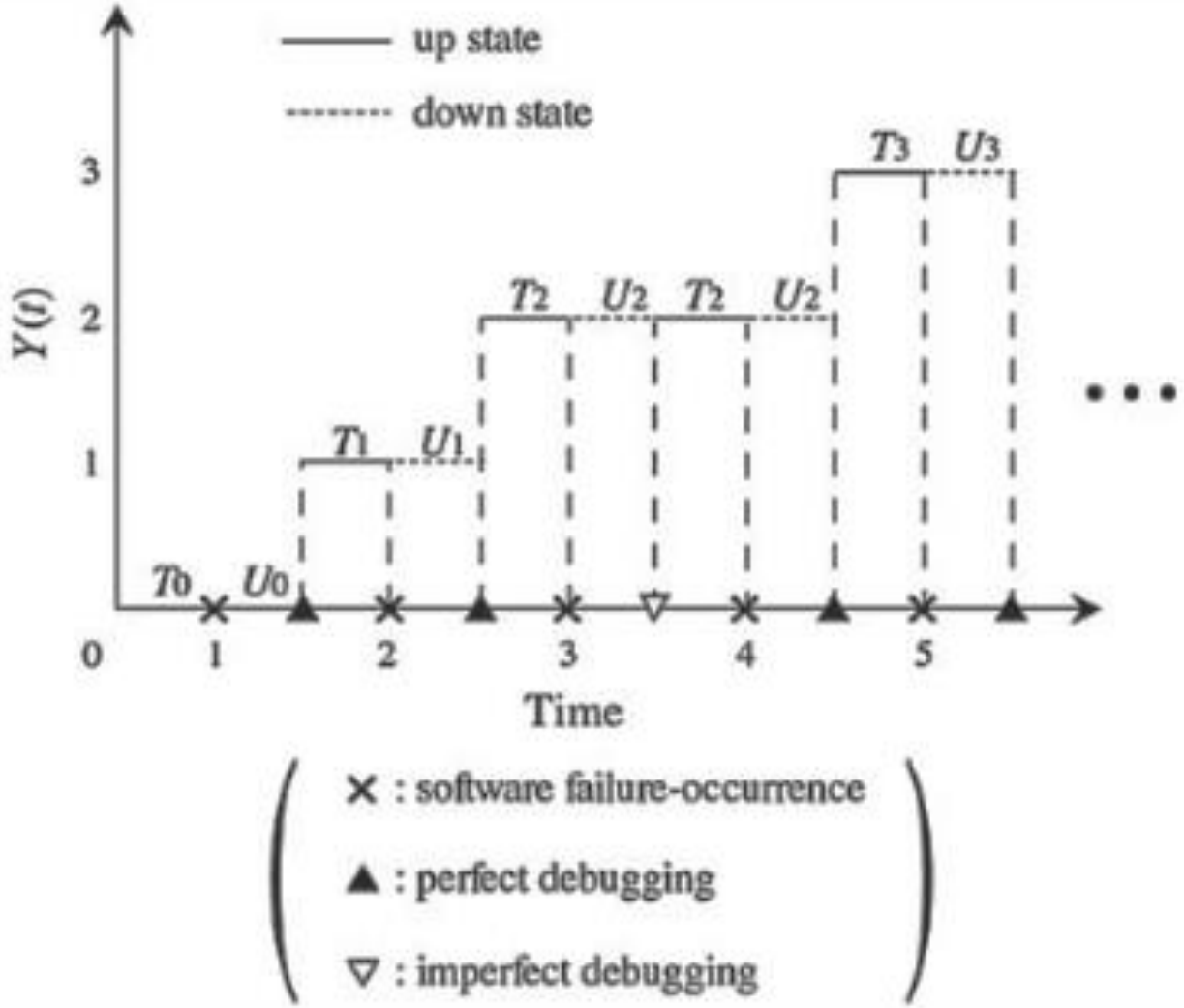


Fig. 1.18 Behavior of restoration rate

Software Availability Measures



Software Availability Measures

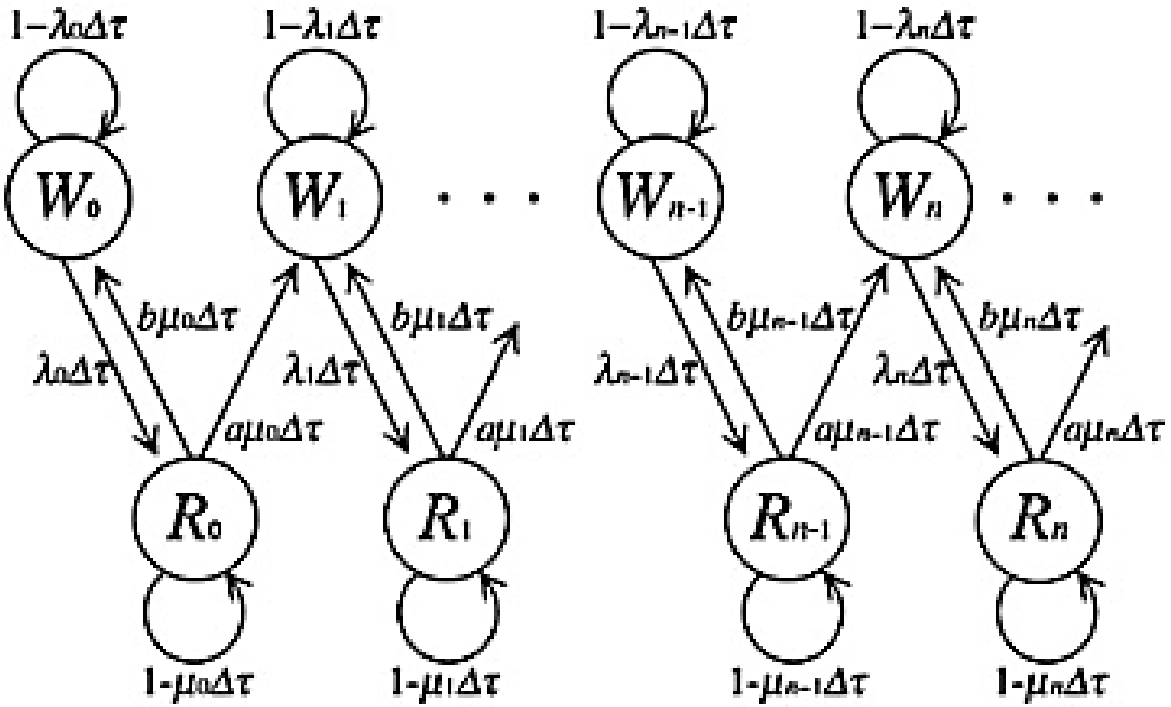


Fig. 1.20 A state transition diagram for software availability modeling

Software Availability Measures

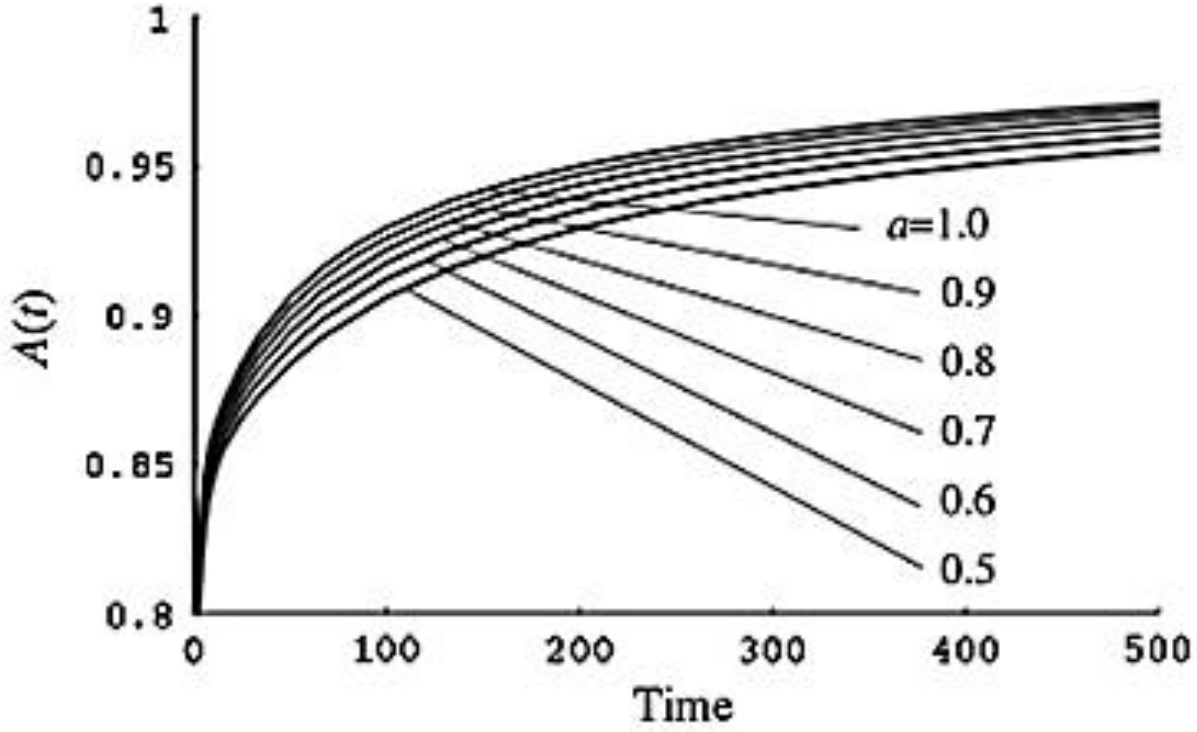


Fig. 1.21 Dependence of perfect debugging rate a on $A(t)$

Software Availability Measures

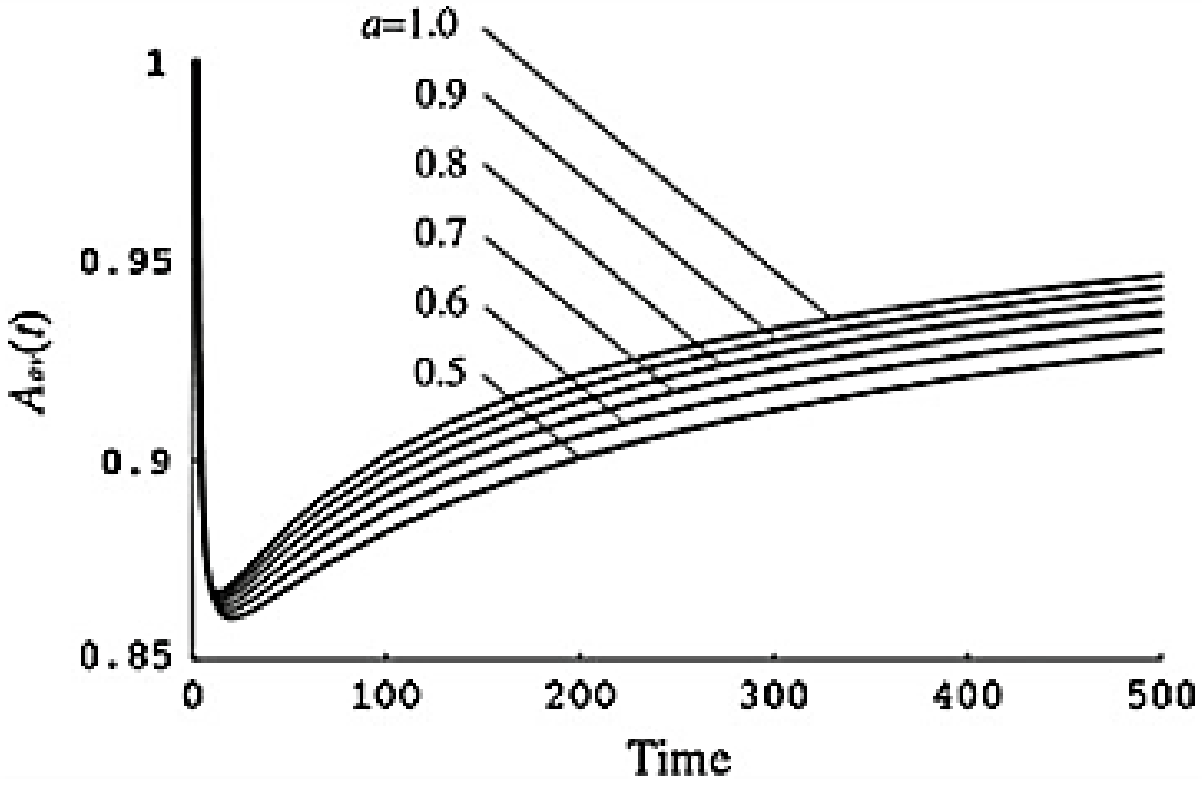


Fig. 1.22 Dependence of perfect debugging rate a on $A_{av}(t)$

Summary

- We have learnt about software availability measures.
- Credits: Yamada book.

Modeling and Simulation

Application of Software Reliability Assessment

Overview

- **Here, we will understand about software reliability assessment**

Application of Software Reliability Assessment

- It is very important to apply the results of software reliability assessment to manage-ment problems with software projects for attaining higher productivity and quality.

Application of Software Reliability Assessment

- **We discuss three software management problems as application technologies of soft-ware reliability models.**

Application of Software Reliability Assessment

- Optimal Software Release Problem
- Statistical Software Testing-Progress Control
- Optimal Testing-Effort Allocation Problem

Summary

- We have learnt about conducting a case study.
- Credits: Paul Fortier book.

Modeling and Simulation

Optimal Software Release Problem

Overview

- **Here, we will understand about optimal software release issues**

Optimal Software Release Problem

- Recently, it is becoming increasingly difficult for the developers to produce highly reliable software systems efficiently.
- Thus, it has been necessary to control a software development process in terms of quality, cost, and release time.

Optimal Software Release Problem

- In the last phase of the software development process, testing is carried out to detect and fix software faults, introduced by human work, prior to its release for the operational use.

Optimal Software Release Problem

- The software faults that cannot be detected and fixed remain in the released software system after the testing phase.
- Thus, if a software failure occurs during the operational phase, then a computer system stops working and it may cause serious damage.

Optimal Software Release Problem

- **If the duration of software testing is long, we can remove many software faults in the system and its reliability increases. However, this increases the testing cost and delays software delivery.**

Optimal Software Release Problem

- In contrast, if the length of software testing is short, a software system with low reliability is delivered and it includes many software faults which have not been removed in the testing phase.
- Thus, the maintenance cost during the operation phase increases.

Optimal Software Release Problem

- It is therefore very important in terms of software management that we find the optimal length of software testing, which is called an optimal software release problem

Optimal Software Release Problem

- These decision problems have been studied in the last decade by many researchers.
- In optimal software release problems we consider both a present value and a warranty period (in the operational phase)

Summary

- We have learnt about optimal software release.
- Credits: Yamada book.

Modeling and Simulation

Optimal Testing-Effort Allocation

Overview

- **Here, we will understand about Optimal Testing-Effort Allocation**

Optimal Testing-Effort Allocation

- We discuss a management problem to achieve a reliable software system efficiently during module testing in the software development process by applying a testing-effort-dependent software reliability growth model

Optimal Testing-Effort Allocation

- We take account of the relationship between the testing-effort spent during the module testing and the detected software faults where the testing-effort is defined as resource expenditures spent on software testing, e.g. manpower, CPU hours, and executed test cases.

Optimal Testing-Effort Allocation

- **The software development manager has to decide how to use the specified testing-effort effectively in order to maximize the software quality and reliability.**

Optimal Testing-Effort Allocation

- That is, to develop a quality and reliable software system, it is very important for the manager to allocate the specified amount of testing-effort expenditure for each software module under some constraints.

Optimal Testing-Effort Allocation

- We can observe the software reliability growth in the module testing in terms of a time-dependent behavior of the cumulative number of faults detected during the testing stage.

Optimal Testing-Effort Allocation

- **Based on the testing-effort dependent software reliability growth model, we consider the following testing-effort allocation problem:**

Optimal Testing-Effort Allocation

- **Based on the testing-effort dependent software reliability growth model, we consider the following testing-effort allocation problem:**

Optimal Testing-Effort Allocation

1. The software system is composed of M independent modules. The number of software faults remaining in each module can be estimated by the model.
2. The total amount of testing-effort expenditure for module testing is specified.
3. The manager has to allocate the specified total testing-effort expenditure to each software module so that the number of software faults remaining in the system may be minimized.

Summary

- We have learnt about Optimal Testing-Effort Allocation.
- Credits: Yamada book.

Modeling and Simulation

Introduction to formal modeling.

Overview

- **Here, we will understand about the “formal modeling”.**

Formal modeling

- Petri nets PNs provide graphical tool and notation method.
- Include arrival rate and service rate.
- Individual state information.
- Contention and concurrency.

Formal modeling

- **Petri Nets have place in computer systems performance assessment.**
- **Between analytical queuing theory and computer simulation.**

Formal modeling

- System state more complete than analytical models.
- But not as simulations.
- Have fundamental theory but act like simulations.
- Single entities, movement and affect effect of the state of the entire system.

Formal modeling

- Petri Nets was introduced in 1966.
- It describes concurrent systems.
- Followed by continuous improvements.
- Timing of transitions.
- Priority of transitions.

Formal modeling

- Types to Token.
- Colors depicting.
- Analysis tools and modelling aid.

Summary

- Here, we understood about the “Testing for Significance of Regression”.
- Credits: Paul Fortier book.

Modeling and Simulation

Basic notation-I

Overview

- **Here, we will understand about the “Basic notation”?**

Basic notation

- Petri Nets provides abstraction and information flow.
- Using four fundamental components.
- Places as Circles.
- Transition as bars.
- Arc as line segments.
- Token as dots.

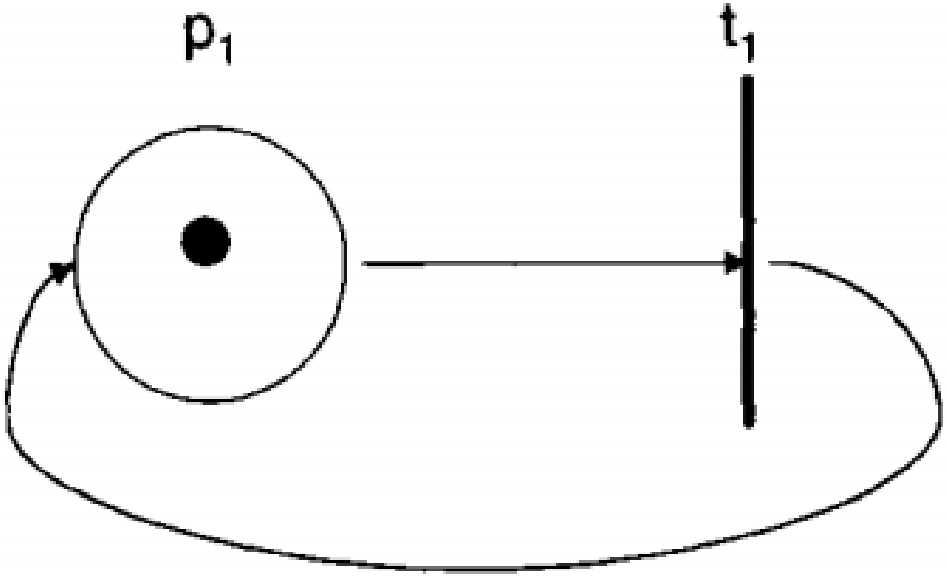
Basic notation

- Places represents useful system components and their state e.g. disk drive.
- Transitions represents events.
- Action of reading an item from disk drive.

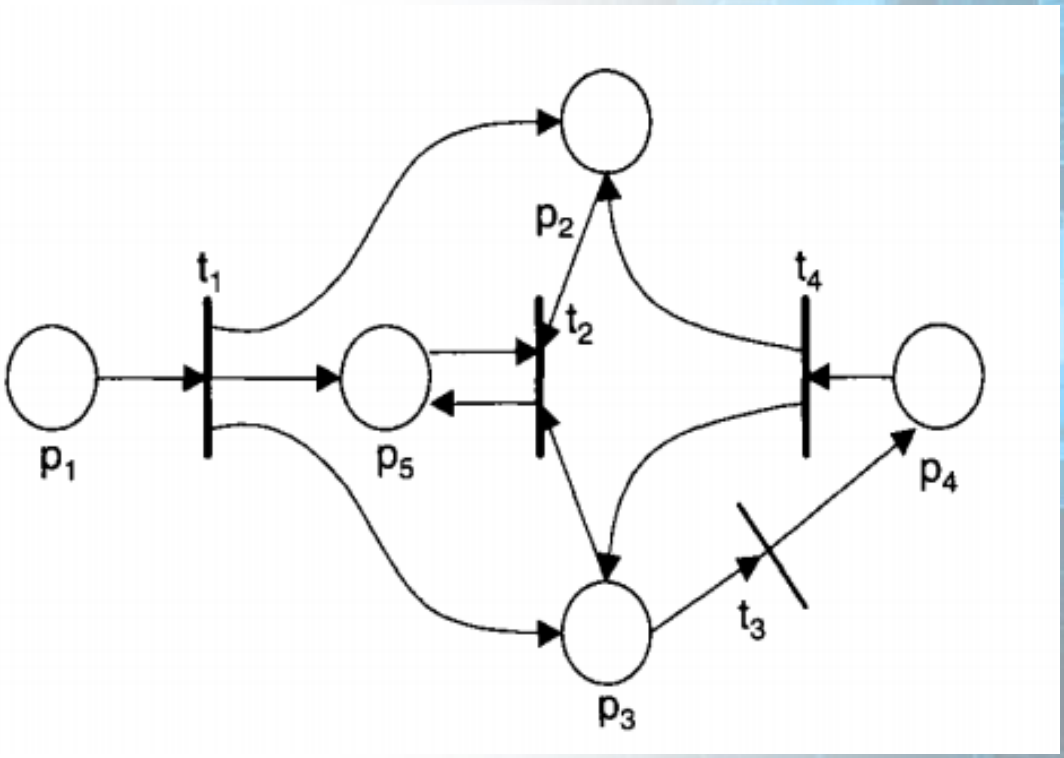
Basic notation

- Arcs represents relationship between transitions and places.
- Proved path for activation of transition.
- Tokens define state of Petri net.

Forever Cycling Petri Net



Petri Net Graphical Representation



Summary

- Here, we understood about Basic notation.
- Credits: Paul Frontier book.

Modeling and Simulation

Basic notation-II

Overview

- **Here, we will understand about the “Basic notation”.**

Basic notation-II

Petri Net Dual

- Petri net model also has dual.
- In dual of petri, transitions are changed to places and places to transitions.
- The input becomes output function and output becomes output function.

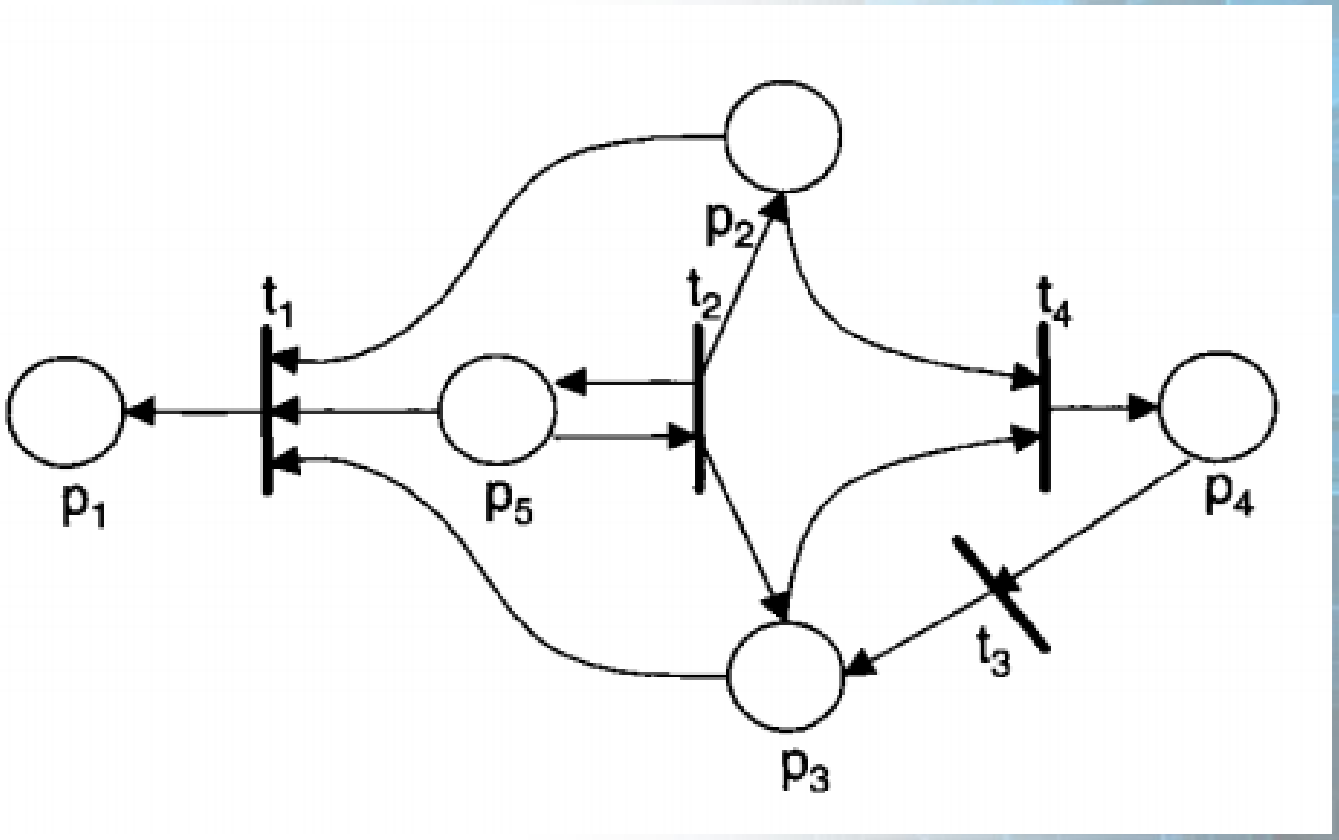
Basic notation-II

Petri Net Inverse

- A petri net model is also defined for inverse.
- Places and transitions are kept same.
- Whereas, input functions switches to output functions.

Basic notation-II

Petri Net Inverse Representation



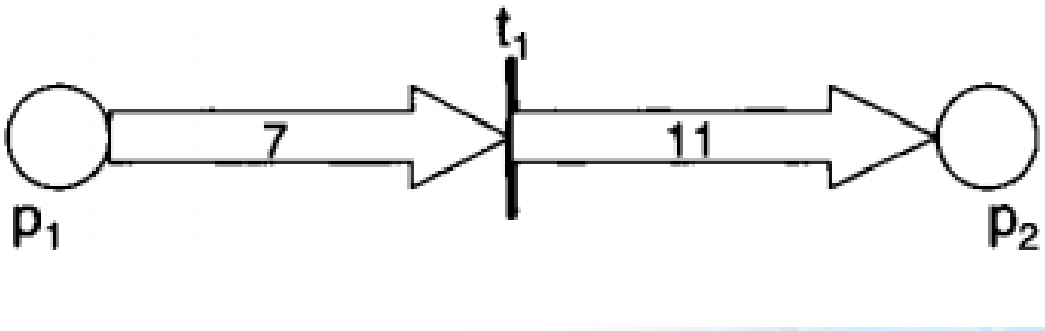
Basic notation-II

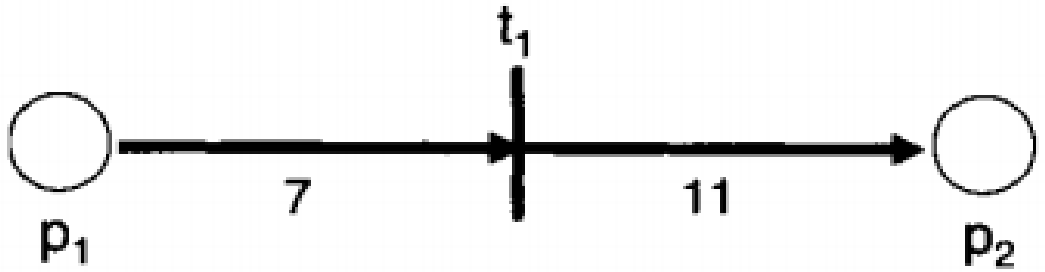
Petri Net Multigraphs

- A petri net model is also defined for Multigraphs.
- It implies there could be several arcs between a single place and a transition.
- Can be represented as:
- Multiple arc as thick arc with numbers inside.
- A bold arc with number attached to it.

Basic notation-II

Basic Petri Net Multigraphs Representation





Basic notation-II

Petri Net State

- A petri net also has a state defined by the cardinality if tokens.
- Represented as function, μ. $\mu : p \rightarrow N$
- μ can be defined as vectors.

$$\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_n),$$

Basic notation-II

Petri Net Multigraphs

- **Where.**

$$n = |P| \text{ and each } \mu_i \in N, i = 0, \ldots, n$$

- **Place marking function and vector relation:**

$$\mu(p_i) = \mu_i.$$

- **Petri net structure:**

$$M = (P, T, I, O, \mu_t)$$

- **It represents state of petri net at time t.**

Summary

- Here, we understood about the “Basic Notaions”.
- Credits: Paul Fortier book.

Modeling and Simulation

Classical Petri Nets-I

Overview

- **Here, we will understand about the “Classical Petri Nets”?**

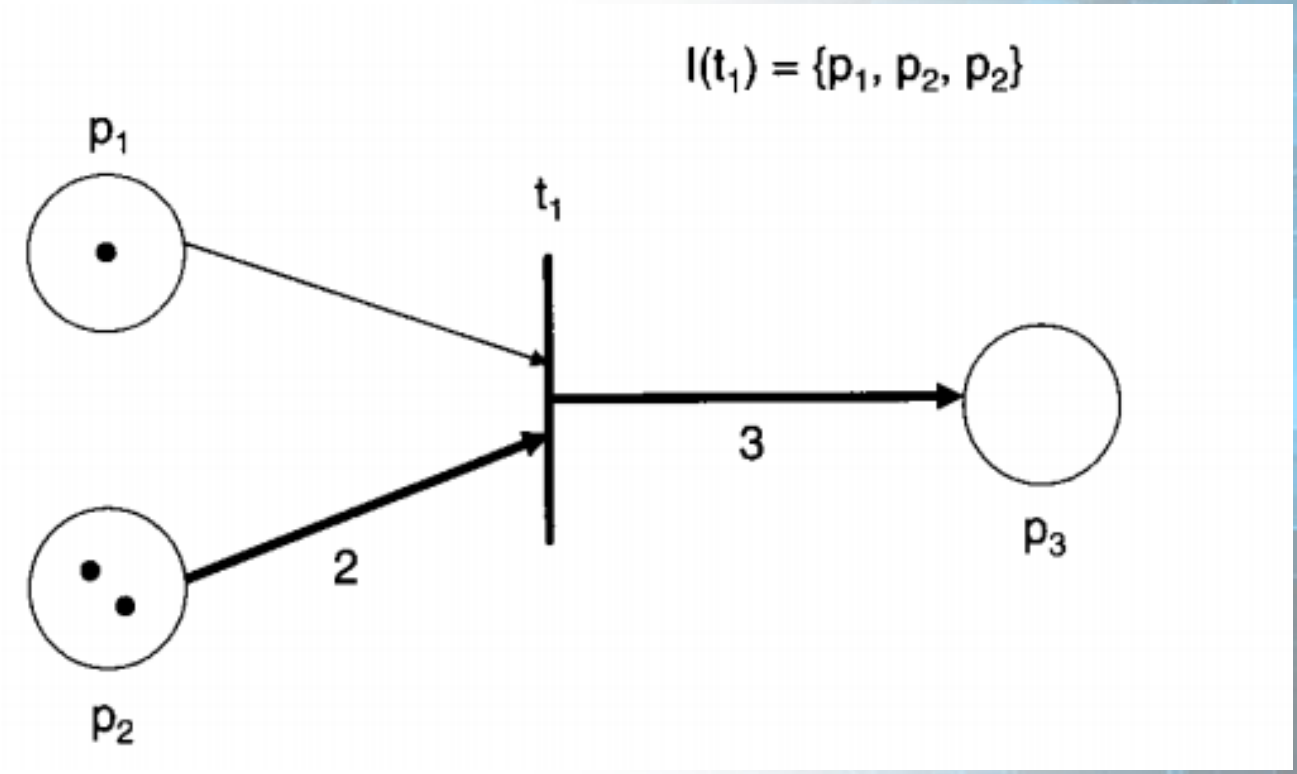
Classical Petri Nets

- By using notations Petri nets can be used in modelling.
- A new petri net state transition notation is required.
- These transitions are helpful in movements.

Classical Petri Nets

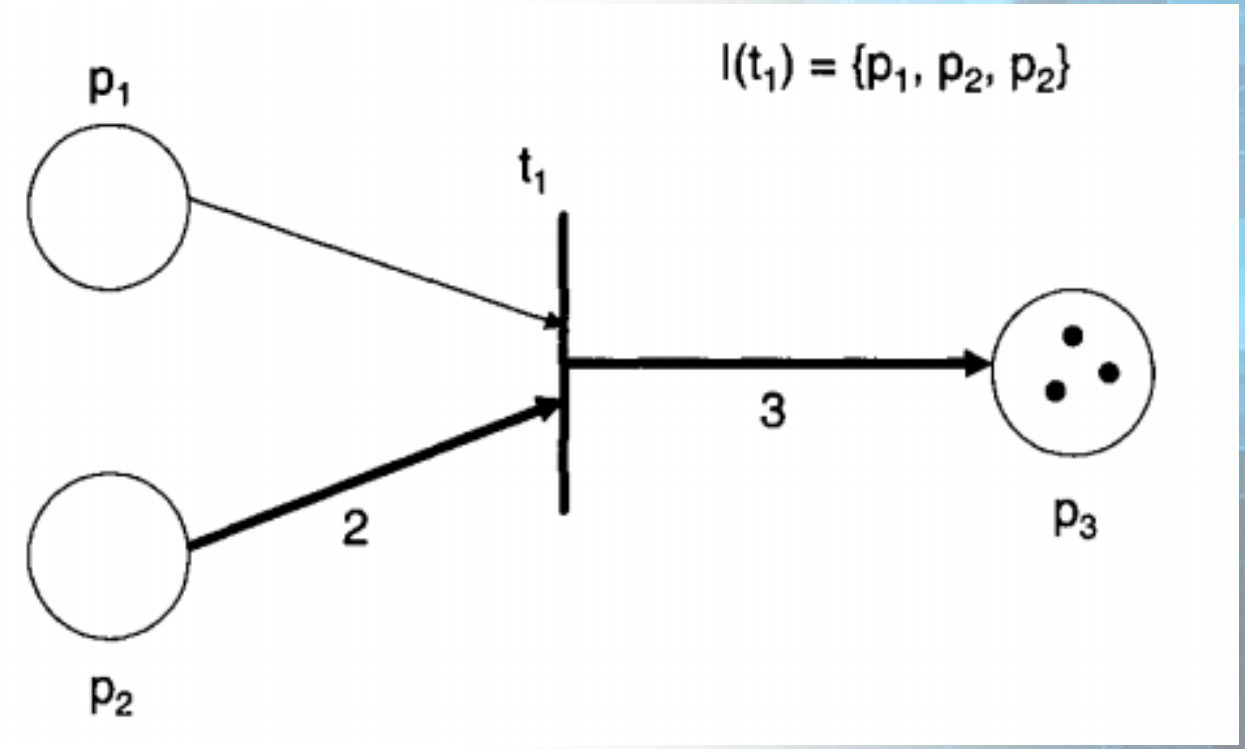
- The exact firing moment can be captured as a clock signal.
- During Clock cycle, all gates start signal execution.
- In petri net all transitions fire once during this cycle.

Classical Petri Nets



Classical Petri Nets

Forever Cycling Petri Net



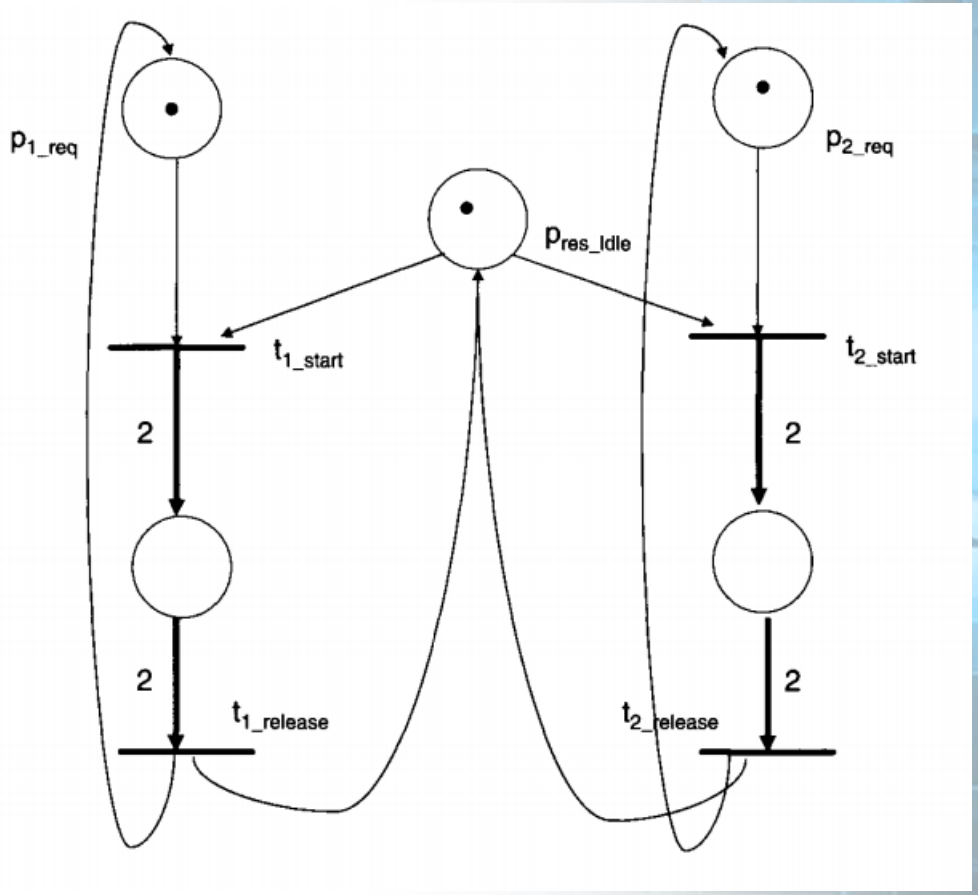
Classical Petri Nets

Petri Net Set Notaion Representation

- Petri nets enables modeling not available through queuing theory.
- Synchronization.
- Conflict.
- Concurrency.
- are not easily defined and modeled by queuing theory.

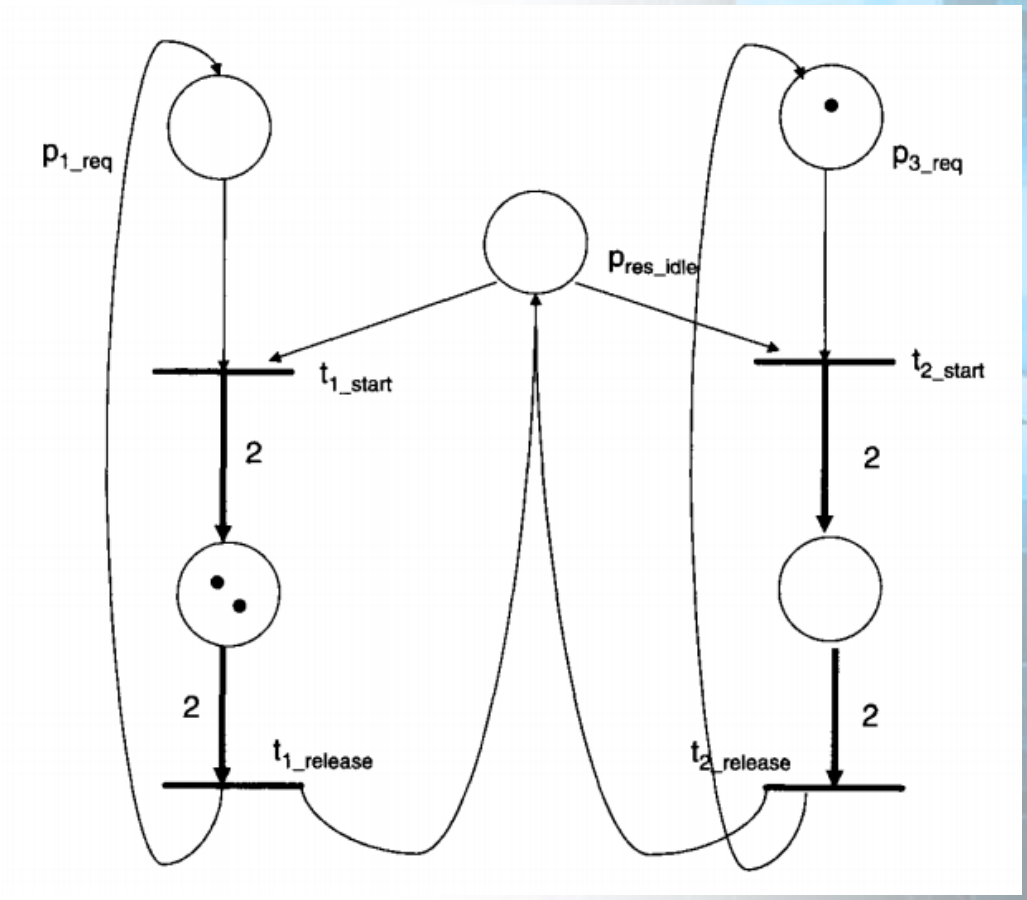
Classical Petri Nets

Resource Sharing



Classical Petri Nets

Allocated Resources



Summary

- Here, we understood about Classical Petri Nets.
- Credits: Paul Frontier book.

Modeling and Simulation

Classical Petri Nets-II

Overview

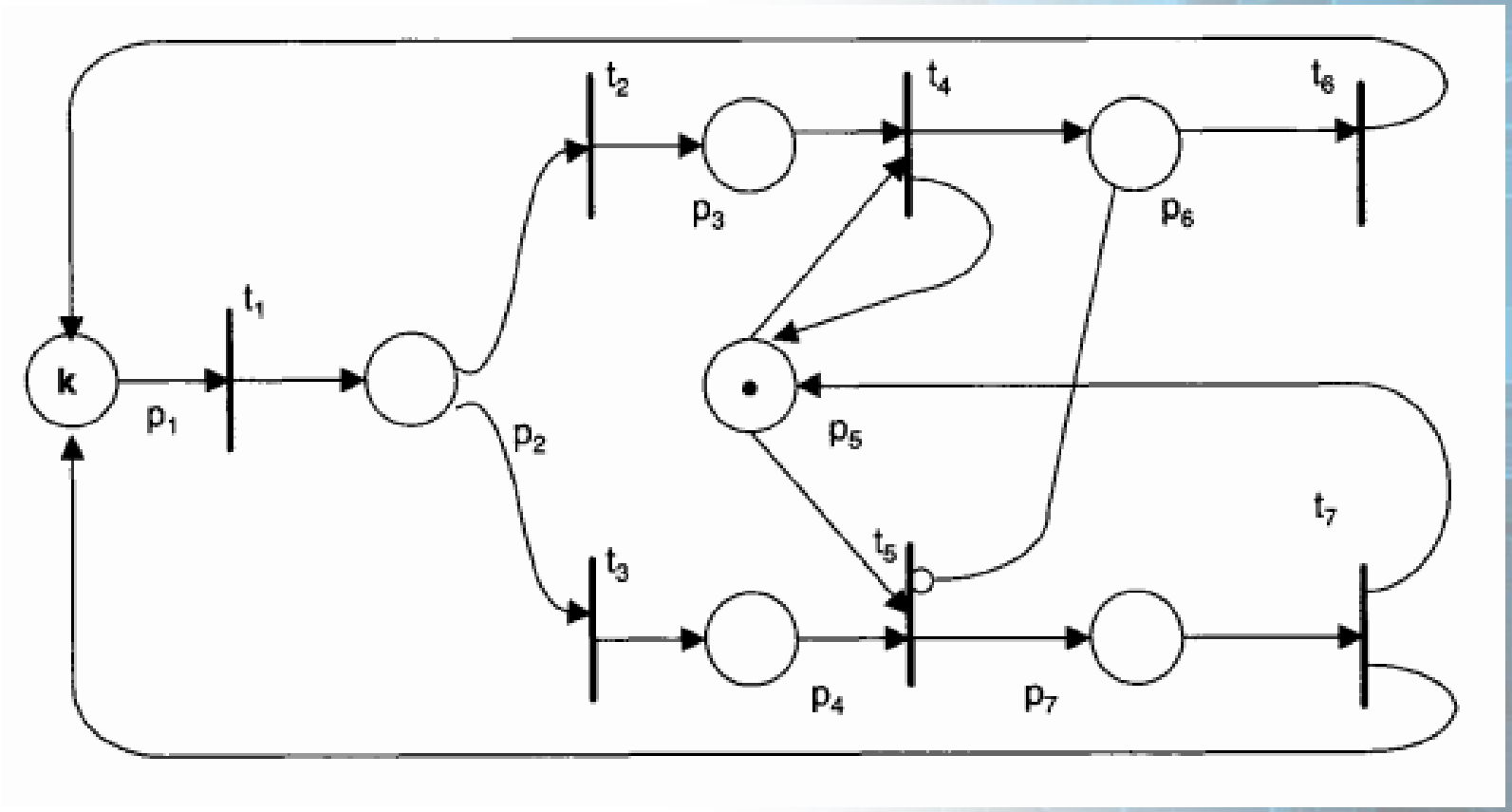
- **Here, We will understand “Classical Petri Nets”.**

Classical Petri Nets-II

Classical Petri Nets

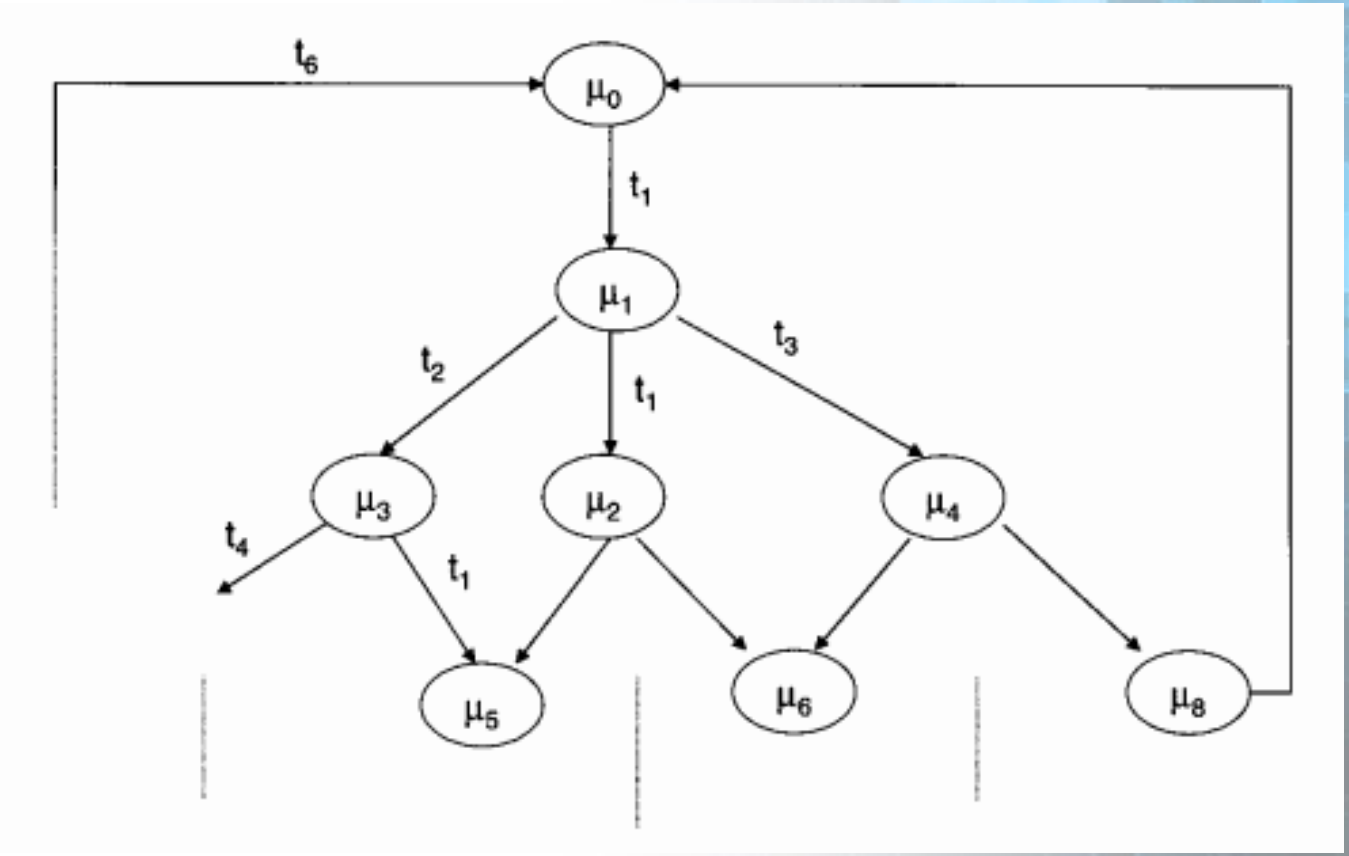
- A Petri net's state, is said to be reachable from some other state, u'.
- There exists some finite number of firings of the Petri net beginning at state u.

Classical Petri Nets-II



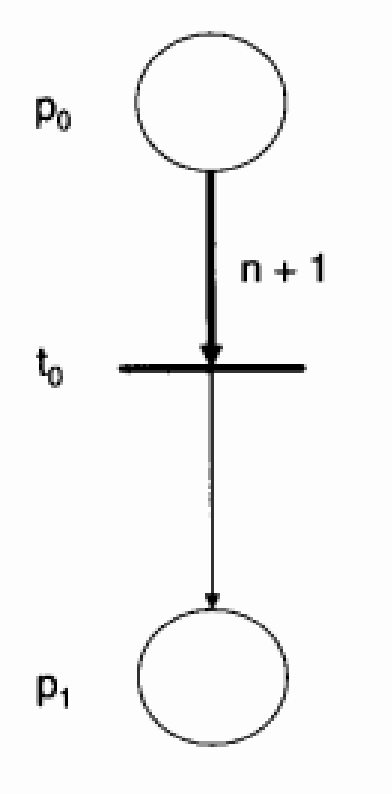
Classical Petri Nets-II

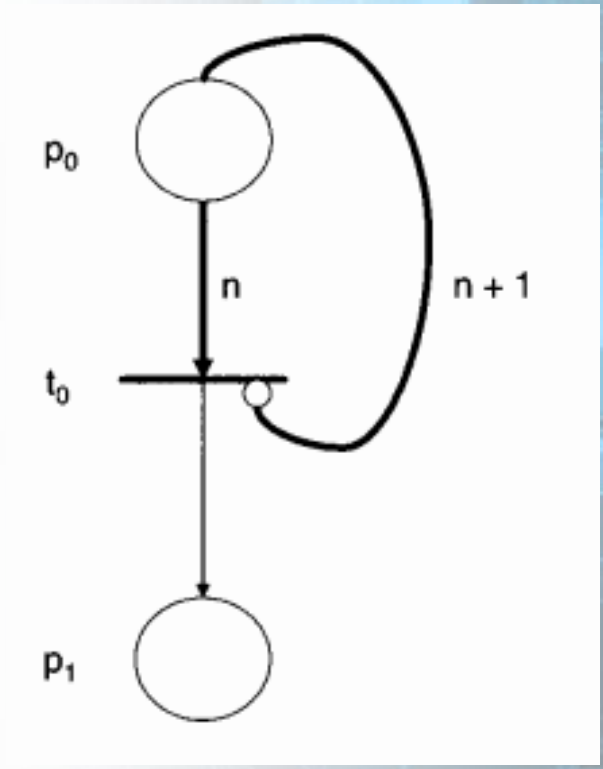
Reachability Graph



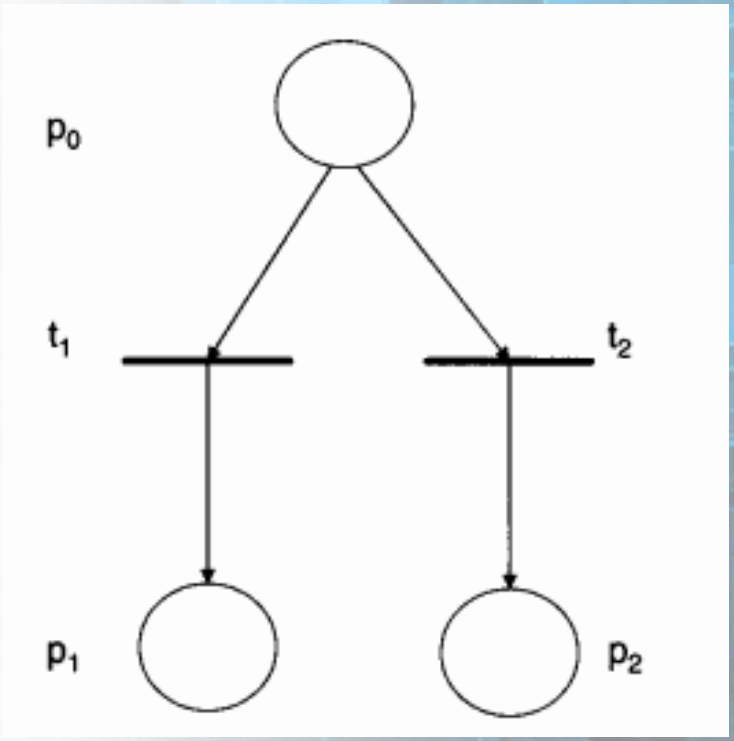
Classical Petri Nets-II

Component Test

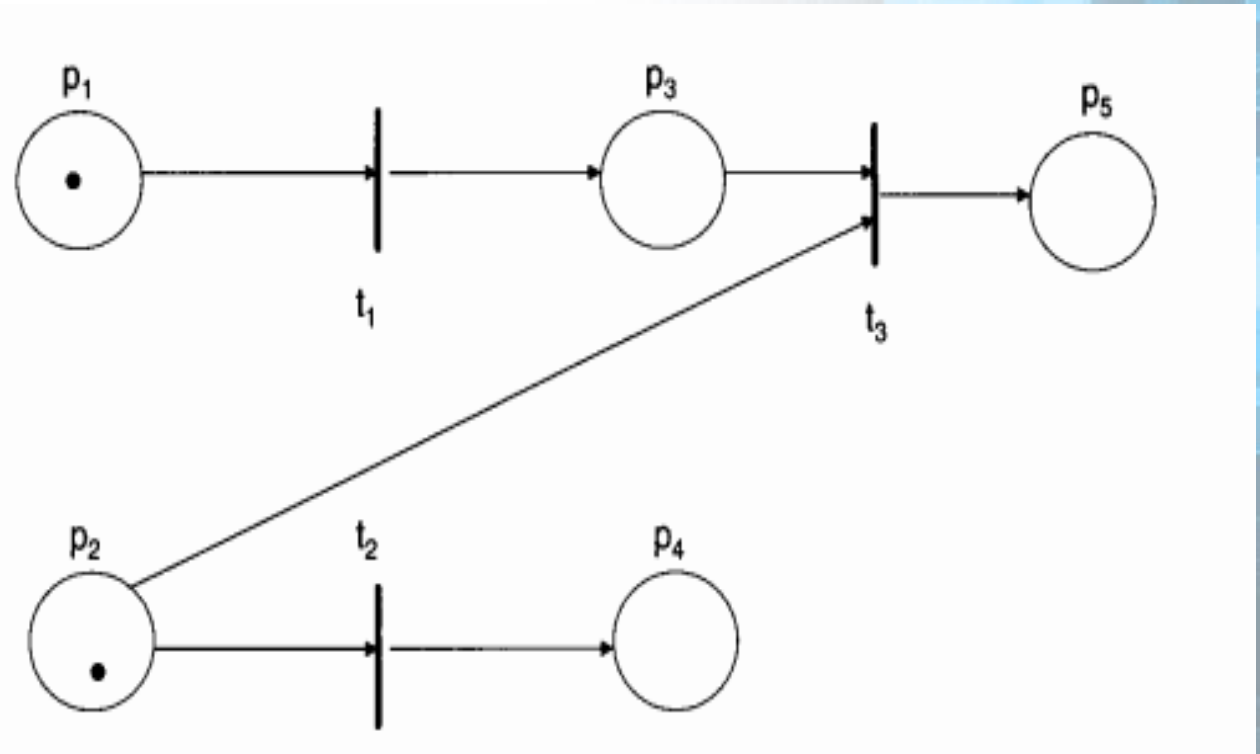




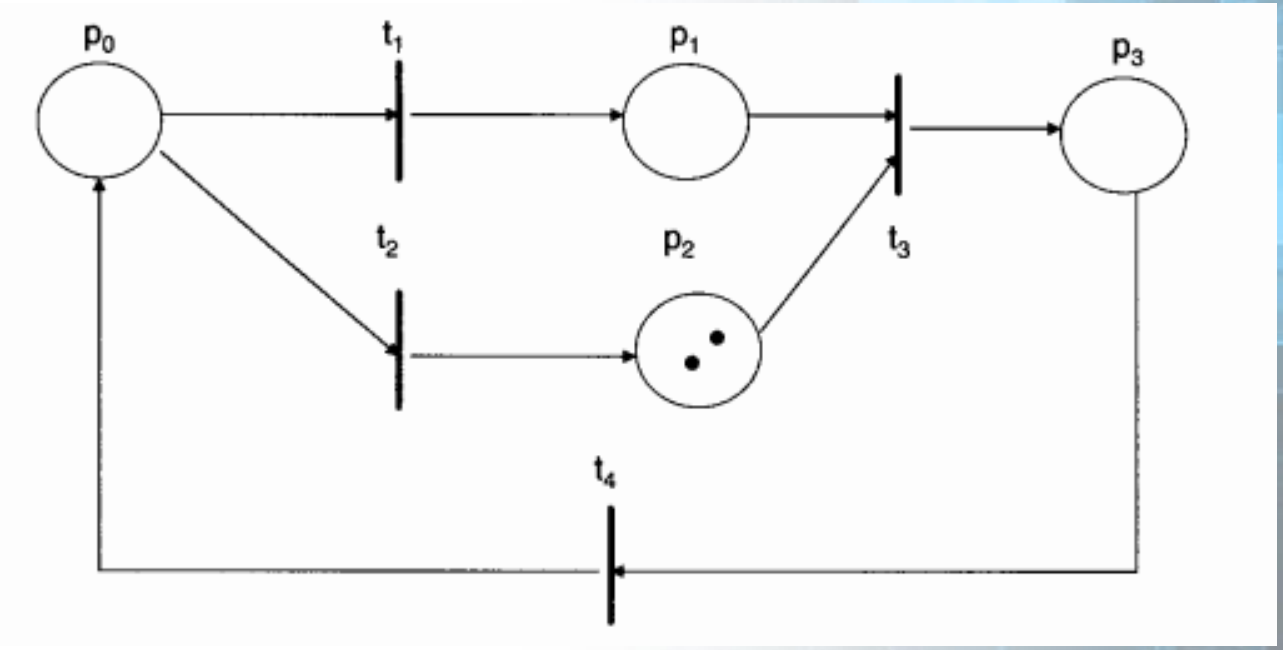
Classical Petri Nets-II



Classical Petri Nets-II



Classical Petri Nets-II



Summary

- Here, we understood about Classical Petri Nets
- Credits, Paul Frontier Book.

Modeling and Simulation

Timed Petri nets I

Overview

- **Here, we will learn about the timed petri nets.**

Timed Petri nets

- Every action takes some time to complete.
- Time addition provides Petri net modeler another powerful tool.

Timed Petri nets

- **Time in Petri net modeling is associated with transitions.**

Timed Petri nets

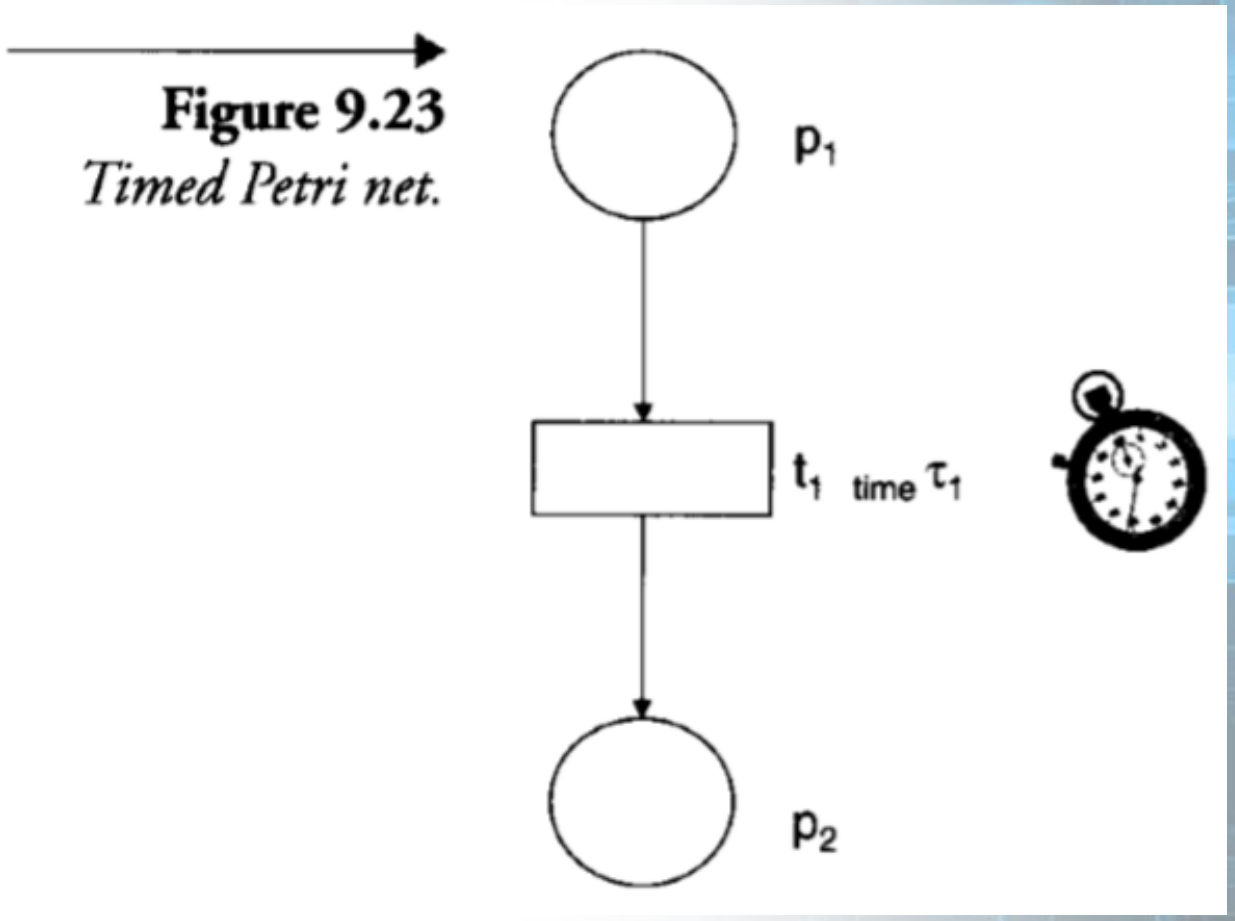


Figure 9.23
Timed Petri net.

Timed Petri nets

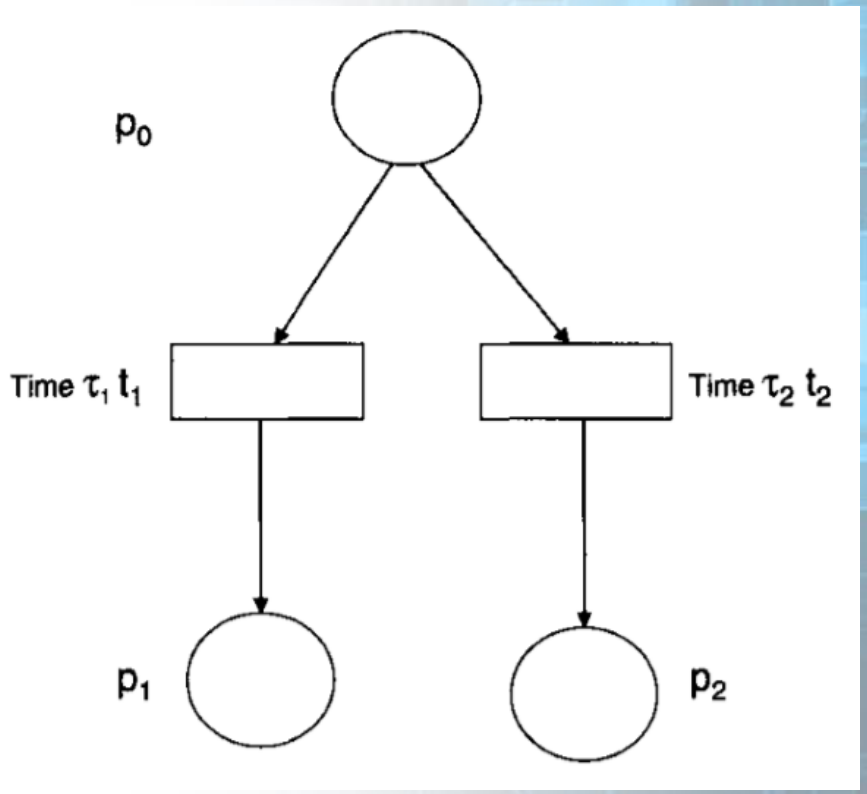


Figure 9.24
Timed Petri net with conflict.

Summary

- Here, we have learnt about the two conditions of the timed Petri nets.
- Credits: Paul Fortier book.

Modeling and Simulation

Timed Petri nets II

Overview

- **Here, we will learn about the timed Petri nets.**

Timed Petri nets

- Some Petri net models have proposed using state transition timing graphs.
- Each state μ is enumerated.
- Time period is set for each state in sequence.

Timed Petri nets

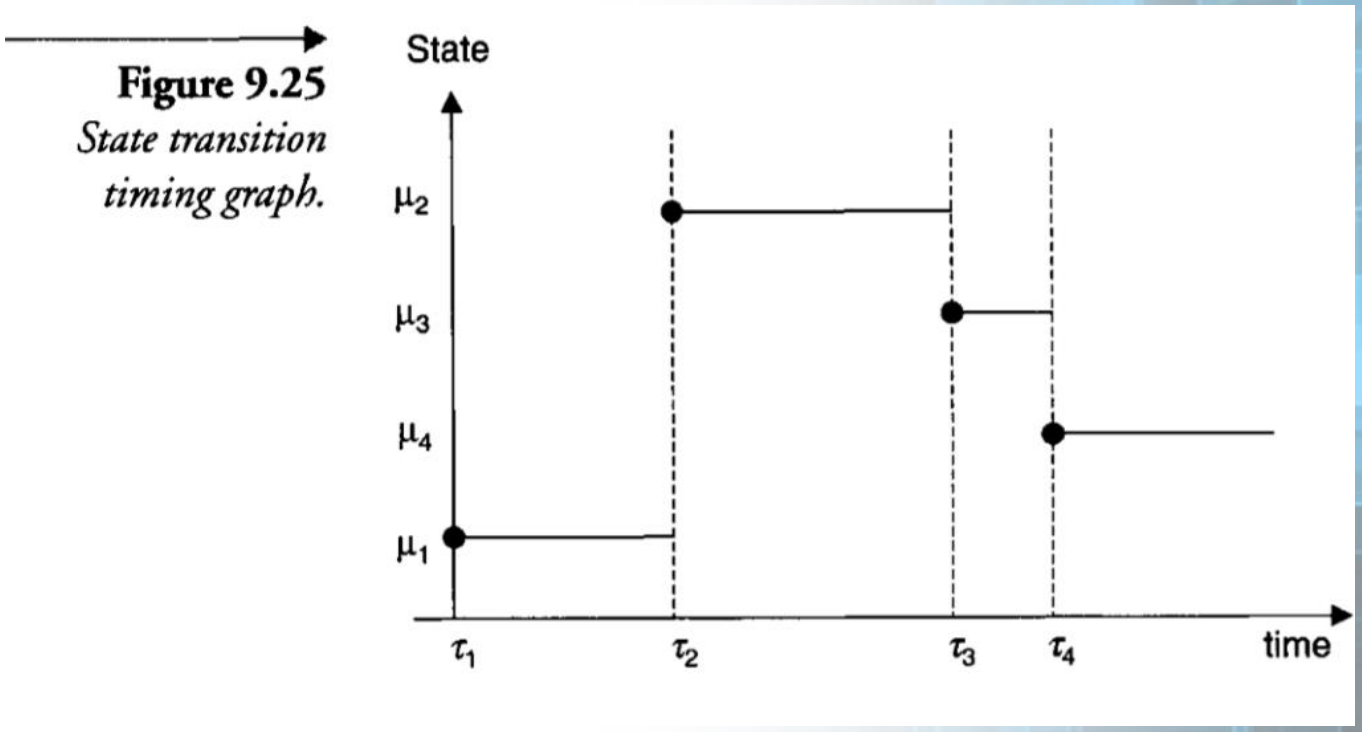


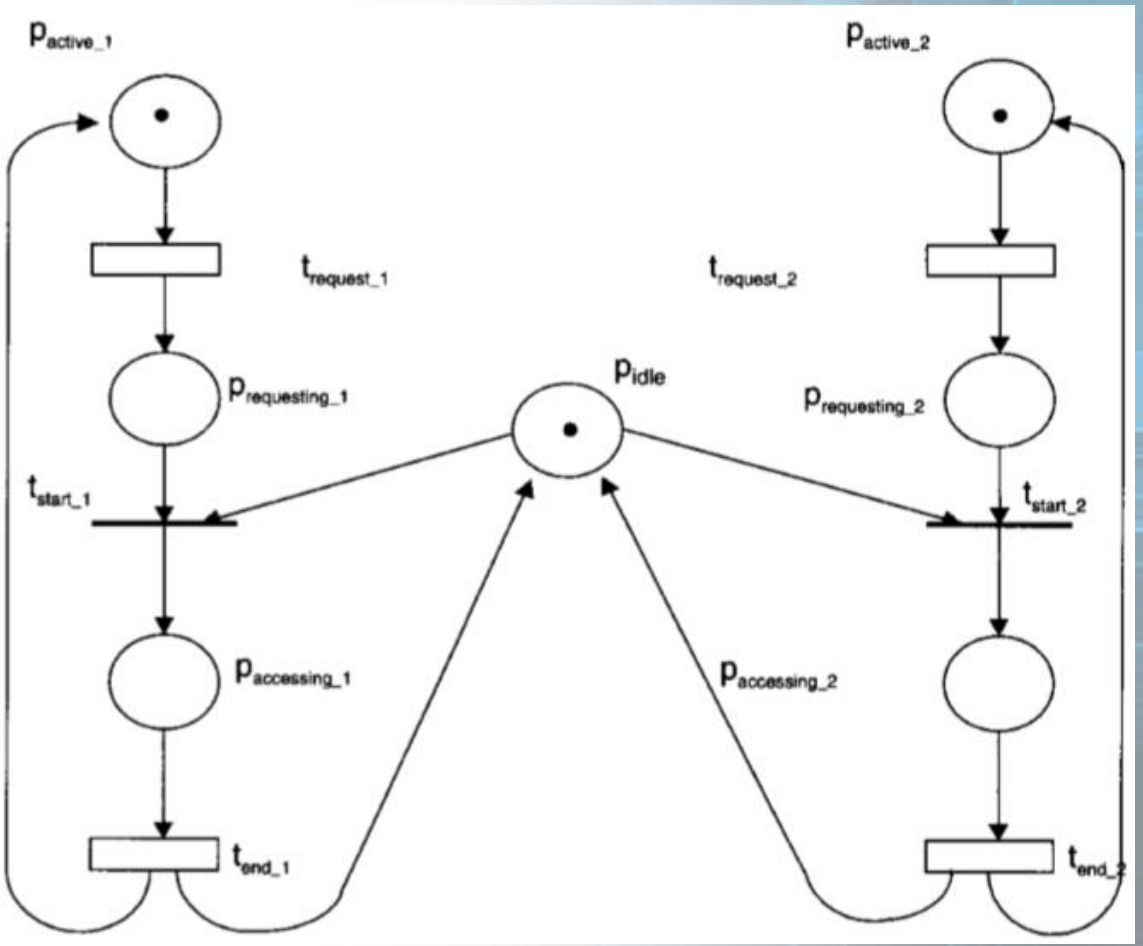
Figure 9.25
State transition timing graph.

Timed Petri nets

- **Time units can be represented using time transition sequences.**
- $[(\tau_1, t_2); (\tau_2, t_2); \ldots; (\tau_j, t_j); \ldots]$

Timed Petri nets

Figure 9.26
Timed Petri net with immediate transitions.



Summary

- We have learnt about the timed Petri nets.
- Credits: Paul Fortier book.

Modeling and Simulation

Priority-based Petri nets

Overview

- **Here, we will learn about the Priority-based Petri nets.**

Priority-based Petri nets

- The Petri net is described by nine tuple.

- $$M = \left(P, T, I, O, H, \Pi, \mathrm{PAR}, Pred, \mu\right)$$

Priority-based Petri nets

- Petri nets are enabled with right amount of tokens.
- No additional transitions area enabled.

Priority-based Petri nets

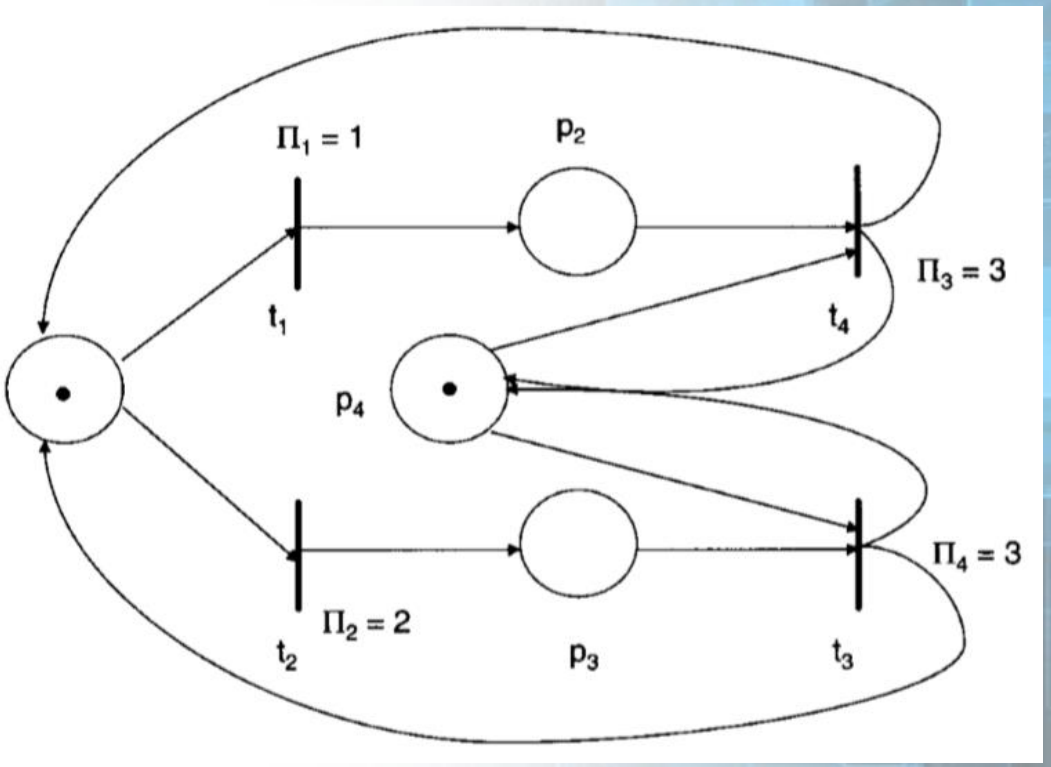
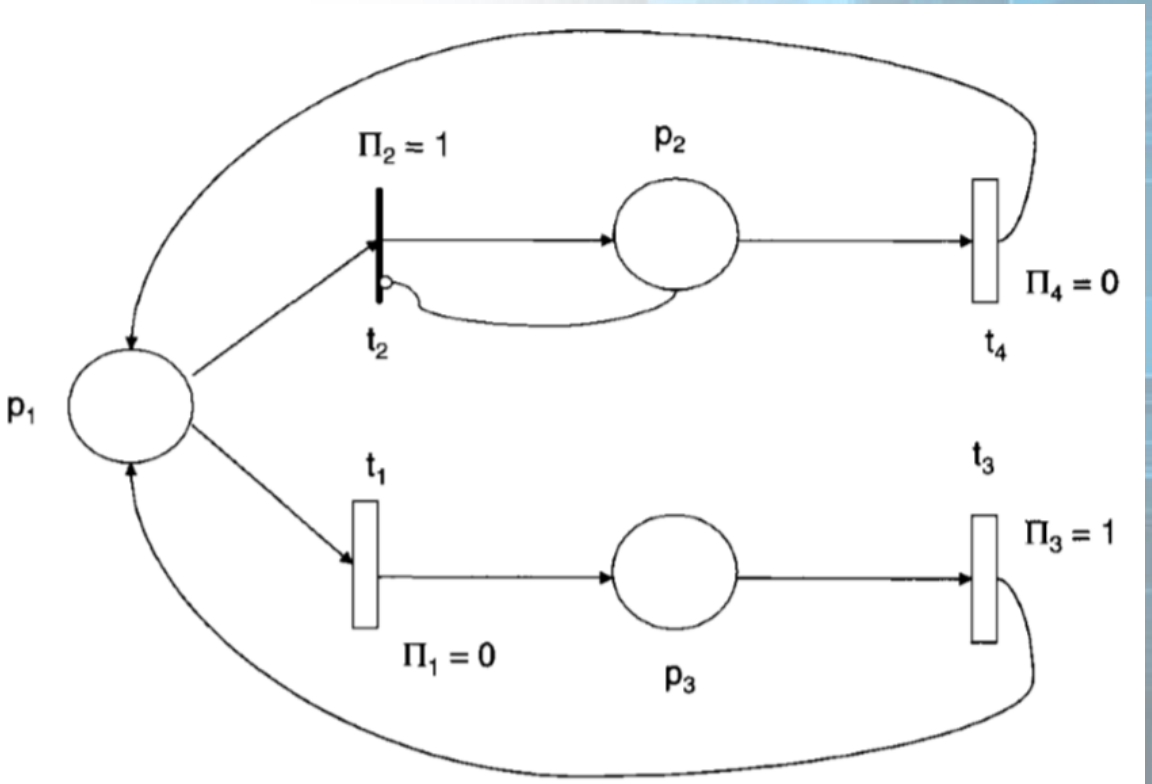


Figure 9.27
Priority-based Petri net.

Priority-based Petri nets

Figure 9.28
Timed and priority net.



Summary

- We have learnt about the priority-based Petri nets.
- Credits: Paul Fortier book.

Modeling and Simulation

Colored Petri nets

Overview

- **Here, we will learn about the colored Petri nets.**

Colored Petri nets

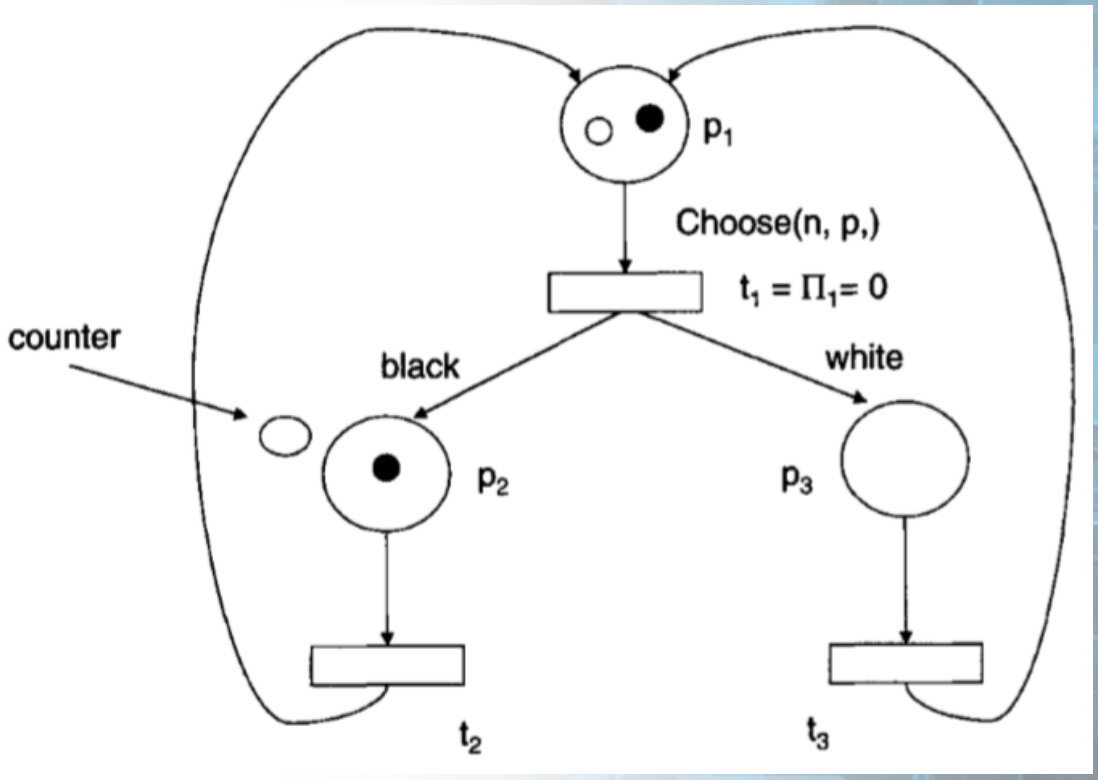
- Different token types.
- Colored tokens are used to represent graphically.

Colored Petri nets

- Cardinality of the token is represented by all colors.
- All definitions of petri nets can be redefined now.
- And firing rules.

Colored Petri nets

Figure 9.29
Generalized Petri net.



Colored Petri nets

- **We can model about any complex condition in systems.**

Summary

- We have learnt about the colored Petri nets.
- Credits: Paul Fortier book.

Modeling and Simulation

Generalized Petri nets

Overview

- **Here, we will learn about the Generalized Petri nets .**

Generalized Petri nets

- Rate or weight transition function.
- $W(t_k\mu)$ must be defined.
- $W(t_k)$ can refer to this when function needs no definition.
- This is rate of transition.

Generalized Petri nets

- To compute time to fire transition an exponential function is used.
- Rate must be define.
- Computed timer value is decremented until 0.

Generalized Petri nets

- **System state changes from one state to another.**

Generalized Petri nets

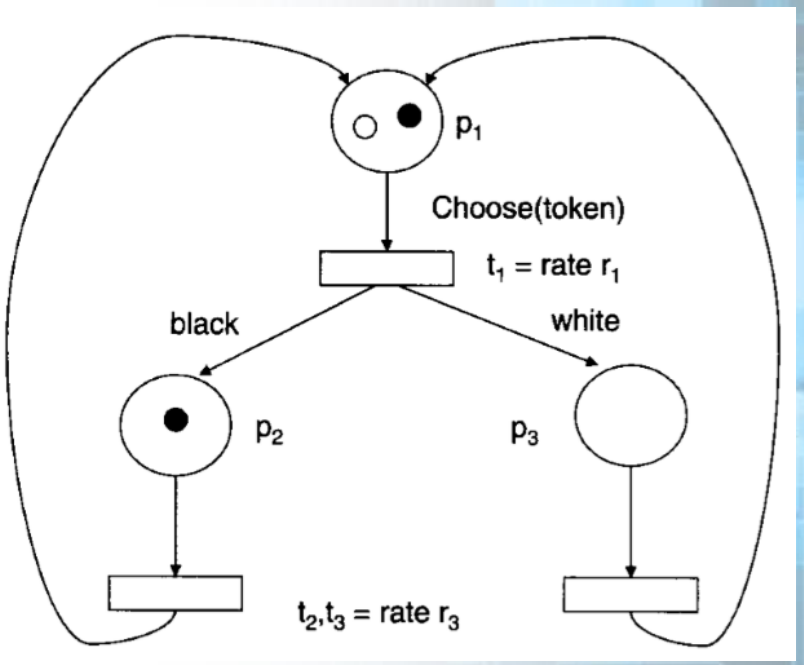


Figure 9.30
Generalized Petri Net.

Generalized Petri nets

- **Each state will not be computed at the same time.**

Generalized Petri nets

- We have learnt about the generalized Petri nets.
- Credits: Paul Fortier book.

Modeling and Simulation

Formal Spec. Using Z

Overview

- **Here, we will understand about formal specification using Z**

Formal Spec. Using Z

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Formal Spec. Using Z

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Formal Spec. Using Z

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Formal Spec. Using Z

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Formal Spec. Using Z

- **Recently, software performance measures such as the possible utilization factors have begun to be interesting for metrics as well as the hardware products.**

Summary

- We have learnt about formal specification using Zed.
- Credits: Online.