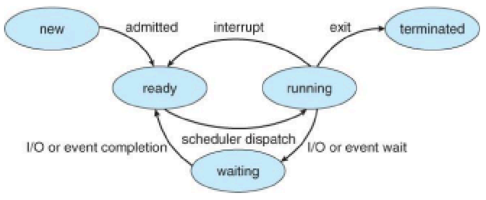
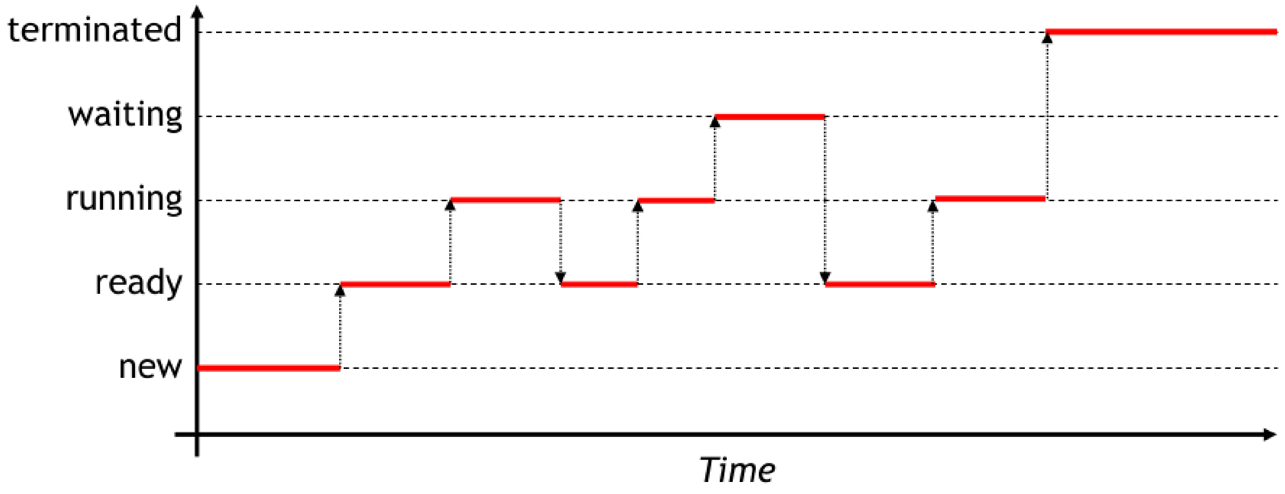
10.State Machine

We generate arrivals and service curves in accordance with the system. The state of a machine can correspond to arrivals, running job, completion etc... it can be the evolution of the process. We can plot how much time the process spend in each state.
We know that a process will stay in a state for a time determined from a random variable. From the states we can derive the performance matrices.

Using the given distributions and the corresponding state machine, a *synthetic trace* of the evolution of the system can be generated.





We can compute the fraction of time similar to the utilization of a system.
The fraction of time spent π_S in a given state s can be determined in a simple way. Let us call T_s the time spent by the system in state s during the time of experiment T. Then:

$$\pi_s = \frac{T_S}{T}$$

Please note that, since random numbers are used, we should always compute confidence intervals, and never rely on simple averages. However, to simplify the discussion, we will only focus on the average measures.

$$\pi_S^{(i)} = \frac{T_S^{(i)}}{T^{(i)}}, \pi_s \in \frac{\sum_{i=1}^{K} \pi_S^{(i)}}{K} \pm \sqrt{\frac{\sum_{i=1}^{K}(\pi_S^{(i)})^2 - K(\sum_{i=1}^{K} \pi_S^{(i)})^2}{K(K-1)}}$$

State machines: analysis

Although seeming simple, the generation and analysis of synthetic traces produced from state machines is not trivial. Several formal ways of presenting and considering state machines in Performance modelling have been developed in the literature. Here, we will give only an informal presentation, to outline the main difficulties and to introduce the techniques that will be considered later in the course.
We try to generate synthetically the trace of data. We have some function that goes from time and for each time will tell which state the system is and this is enough to store each time the system moves between states and encode how much time it spent in each state.
The general high-level algorithm to produce a trace can then be the following:

```
s = s0;             % Variable s contains the current state
t = 0;              % Variable t contains the current time
i = 0;              % Variable i contains the index of the current output
couple
while t < T do      % Until the end of trace time T
        if s = 1 then
                ns = newstateFromStateMachine…
                dt = durationFromDistribution…
        end
        if s = 2 then
        …
        i = i + 1;
        s = ns;                          % Move to next state
        t = t + dt;                      % Increase time
        Trace(i,:) = [t, s];             % Store in the trace
end
```

Two variables s and t are used to hold the current state, and the current time

Two variables s and t are used to hold the current state, and the current time.

```
s = s0;                     % Variable s contains the current state
t = 0;                      % Variable t contains the current time
i = 0;                      % Variable i contains the index of the current output couple
while t < T do              % Until the end of trace time T
     if s = 1 then
               ns = newstateFromStateMachine…
               dt = durationFromDistribution…
     end
     if s = 2 then
     …
     i = i + 1;
     s = ns;                         % Move to next state
     t = t + dt;                     % Increase time
     Trace(i,:) = [t, s];            % Store in the trace
end
```

The main loop is repeated until the target time T is reached.

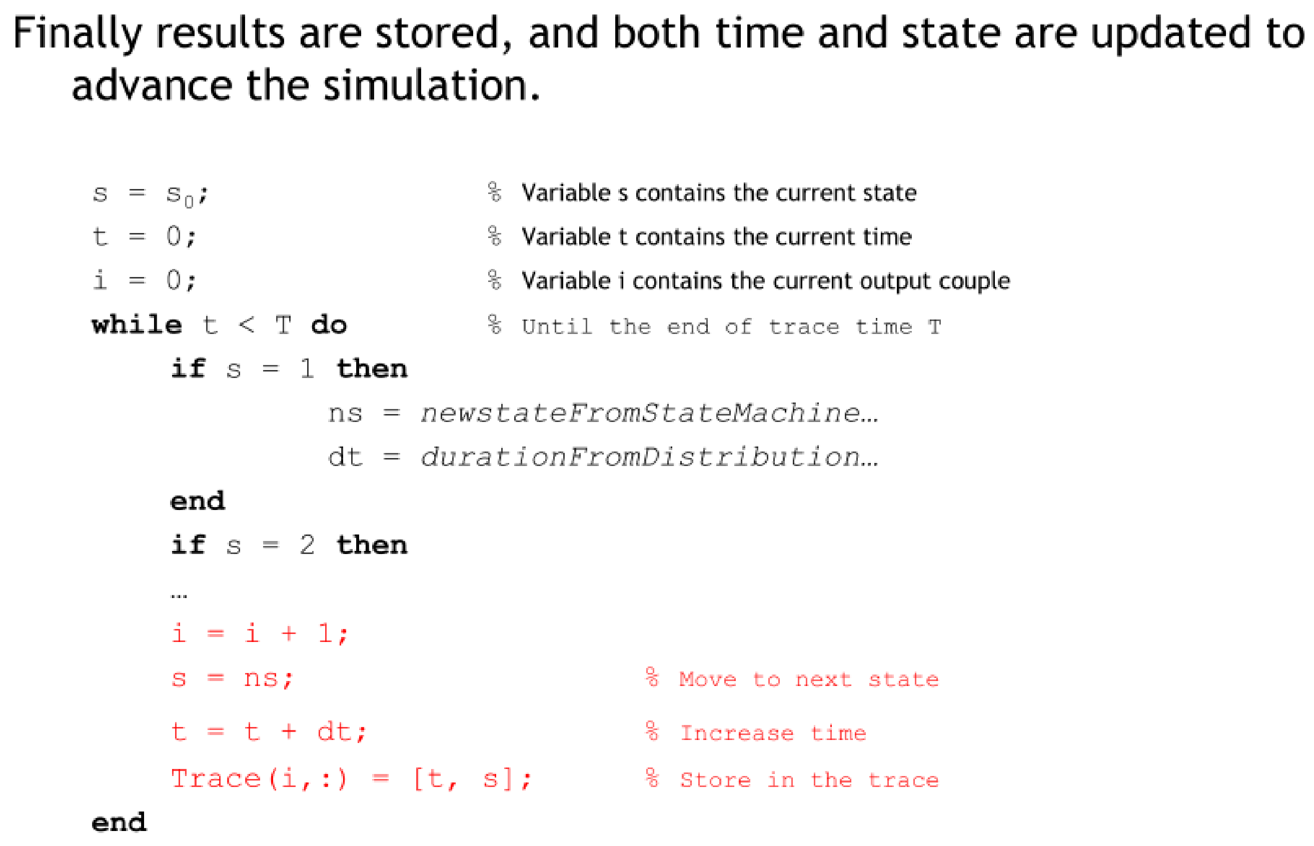
```
s = s0;                     % Variable s contains the current state
t = 0;                      % Variable t contains the current time
i = 0;                      % Variable i contains the index of the current output couple
while t < T do              % Until the end of trace time T
     if s = 1 then
               ns = newstateFromStateMachine…
               dt = durationFromDistribution…
     end
     if s = 2 then
     …
     i = i + 1;
     s = ns;                         % Move to next state
     t = t + dt;                     % Increase time
     Trace(i,:) = [t, s];            % Store in the trace
end
```

Depending on the current state, the next state n_s and the time d_t after which state change occurs, are determined:

```
s = s0;                  % Variable s contains the current state
t = 0;                   % Variable t contains the current time
i = 0;                   % Variable i contains the index of the current output couple
while t < T do           % Until the end of trace time T
    if s = 1 then
            ns = newstateFromStateMachine…
            dt = durationFromDistribution…
    end
    if s = 2 then
    …
    i = i + 1;
    s = ns;                      % Move to next state
    t = t + dt;                  % Increase time
    Trace(i,:) = [t, s];         % Store in the trace
end
```

The actual way in which `ns` and `dt` are determined, depends on the particular state machine we are trying to reproduce:

```
s = s0;                  % Variable s contains the current state
t = 0;                   % Variable t contains the current time
i = 0;                   % Variable i contains the current output couple
while t < T do           % Until the end of trace time T
    if s = 1 then
            ns = newstateFromStateMachine…
            dt = durationFromDistribution…
    end
    if s = 2 then
    …
    i = i + 1;
    s = ns;                      % Move to next state
    t = t + dt;                  % Increase time
    Trace(i,:) = [t, s];         % Store in the trace
end
```



Finally results are stored, and both time and state are updated to advance the simulation.

```
s = s_0;                    % Variable s contains the current state
t = 0;                      % Variable t contains the current time
i = 0;                      % Variable i contains the current output couple
while t < T do              % Until the end of trace time T
    if s = 1 then
            ns = newstateFromStateMachine…
            dt = durationFromDistribution…
    end
    if s = 2 then
    …
    i = i + 1;
    s = ns;                         % Move to next state
    t = t + dt;                     % Increase time
    Trace(i,:) = [t, s];            % Store in the trace
end
```

There are several alternatives possible to select the next state.
Here we will present the main ones:

- Sequence
- Choice (Next time)
- Race (Next time)
- Concurrency (Next time)

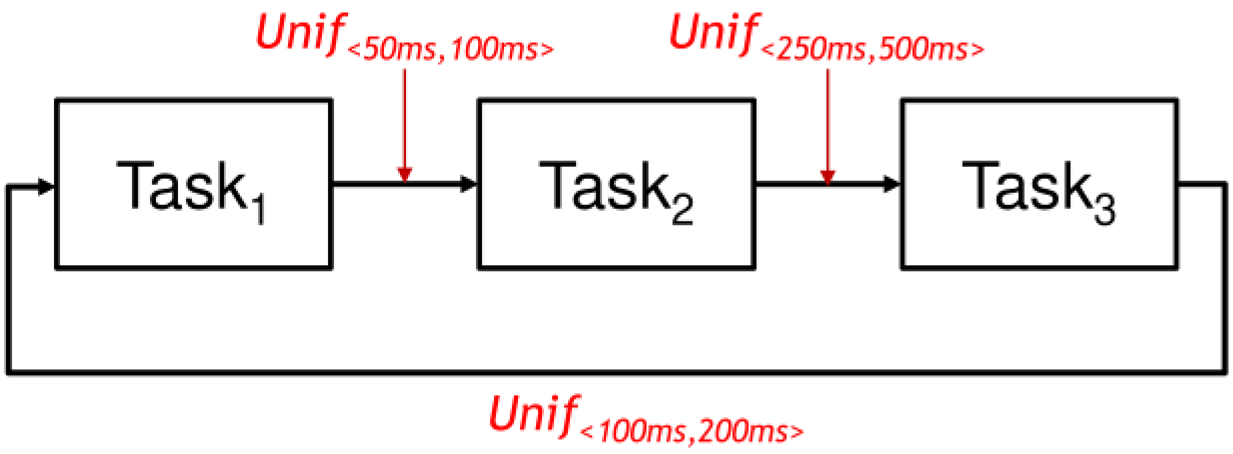
Sequence

Consider and embedded system that shares the CPU running three tasks in a cyclic pattern. Each task has associated the time span where it can run: this is different for each task, depending on their function and priority. In particular, we have measured that length of each task can be modeled with a uniform distribution, with different extents.
We would like to compute:

- Probability that a given task is in execution
- Average time between two executions of the same task.
  The task graph is already a state machine. In this case, things are very simple since, in each state, there is a unique successor.

```
if s = 1 then
ns = 2;
dt = GenUnform(50,100);
end
if s = 2 then
```

```
ns = 3;
dt = GenUnform(250,500);
end
if s = 3 then
ns = 1;
dt = GenUnform(100,200);
end
```



```
CurrRountTime = 0; RoundTimeIndex = 1; RTT = [];
Ts1 = Ts2 = Ts3 = 0;
…
if s = 1 then
ns = 2;
dt = GenUnform(50,100);
Ts1 += dt;
CurrRoundTime += dt;
end
if s = 2 then
…
Ts2 += dt;
CurrRoundTime += dt;
end
if s = 3 then
…
Ts3 += dt;
CurrRoundTime += dt;
RRT(RoundTimeIndex) = CurrRoundTime;
CurrRoundTime = 0;
RoundTimeIndex ++;
end
…
Ps1 = Ts1 / T;
Ps2 = Ts2 / T;
```

```
Ps3 = Ts3 / T;
AvgRT = mean(RTT);
```