Dining Philosophers

Example

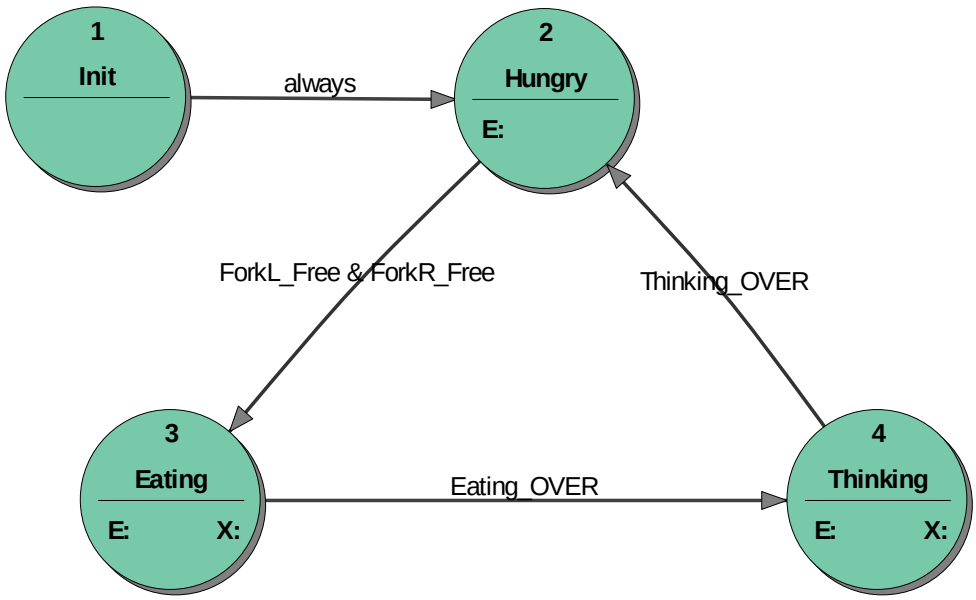
There are some variants of this problem. One of them reads:

“There are five philosophers sitting at a round table who do nothing but think and eat. Between each philosopher there is a single fork. In order to eat, a philosopher must have both forks”.

This problem is used to discuss multi-process synchronisation problems, like deadlocks and starvation. You get these problems if you put some restrictions on the way how a philosopher grabs for a fork, for instance first on the right and than waits for the fork on the left. But today I do not want to model synchronisation problems – so I realize the working system where a philosopher starts eating if both forks at his sides are free. As this example has been invented as a Christmas gift - study starvation would not be a proper topic.

Actually, I took the wording from http://www.codeproject.com/csharp/FSMdotNet.asp?target=state%7Cmachine&select=691061&df=100&forumid=29430&fr=16.5#xx691061xx where I took part in a discussion about state machines. You may find it interesting to compare the effort you need if you code something and using a ready-made execution system like stateworks.

A philosopher's behavior is simulated by a state machine represented by the following state transition diagram:



The eating time and thinking time are defined by separate timers. The forks are represented by Xda objects.

To simulate the problem we need than 5 state machines, one for each philosopher. The state machines are not a system of state machines; they are just 5 separate state machines. The dependencies among philosophers (state machines) come into being as they use common forks (a left fork of one philosopher is the right fork of his neighbour).

Running the example

When you install the StateWORKS Studio you will find the DiningPhilosophers project in the folder ..\Project\Examples-Web\TrafficLight. You may try it using StateWORKS development tools. You may run the SWLab with the DiningPhilosophers example and monitor the system using SWMon and/or SWTerm. The system uses digital outputs to indicate thinking philosophers: if a Do is *on* the philosopher thinks, otherwise he eats or is hungry. In SWMon you may change the timeout values for thinking and eating but I can assure you that for any combination of timeout values no philosopher will starve (well, assuming that you do not use very large values).