State Transition Systems

We would like to take this opportunity to briefly discuss state transition systems. Unlike the system architectures that are presented in this course, and describe the design a system, a state transition system is a concept used to describe the behavior of a system. More precisely, all potential behaviors of a system. Even if a specific system state has a low probability of being seen, a state transition system will still describe how that state can be arrived at if it is an expected behavior of the system.

State Transition Systems

State transition systems are a complex topic in computer science and they play an important role in helping software architects and software engineers understand how different states can be achieved. The goal of this reading is to provide a coarse overview of what a state transition system is, and how they are used, in order to illustrate why they are important.

Let's start by defining the terminology that are used when discussing state transition systems.

- State. The information that a system remembers defines its state. For example, a queue system can be in a number of different "states": an empty state, a queued state, or a full state.
- Transition. A transition is used to describe the change of a system from one state to another. A single system state can have multiple transitions; majority of systems today will have multiple transitions branching from one system state. This makes behaviors non-deterministic, since we cannot predict what the next state of the system will be.
- Behavior. The behavior of a system describes what the system will do when exposed to a condition, which can vary from timed system events to user input.

State transition systems can be labelled or unlabelled.

An unlabelled state transition system is defined as a set of state, S, and transition, →, pairs that are used to describe the system's behavior. If p and q are two different states in S, then the transition between them is depicted by (p → q)

A labelled state transition system simple includes a set of labels, ~, with addition to the state-transition pair. Given the same states p and q in S, then the transition between the two states is shown as (p ⇝ q).

Unlabelled state transitions can be used for systems where the transition between all states are the same. For example, your system could respond to a single button press that simply transitions the system sequentially from one state to the next.

As you can see, state transition systems can be thought of as directed graphs. Each node of the graph represents the set of states, and each edge of the graph represents the set of transition. We can determine how a system reaches a specific state by simply traversing the graph.

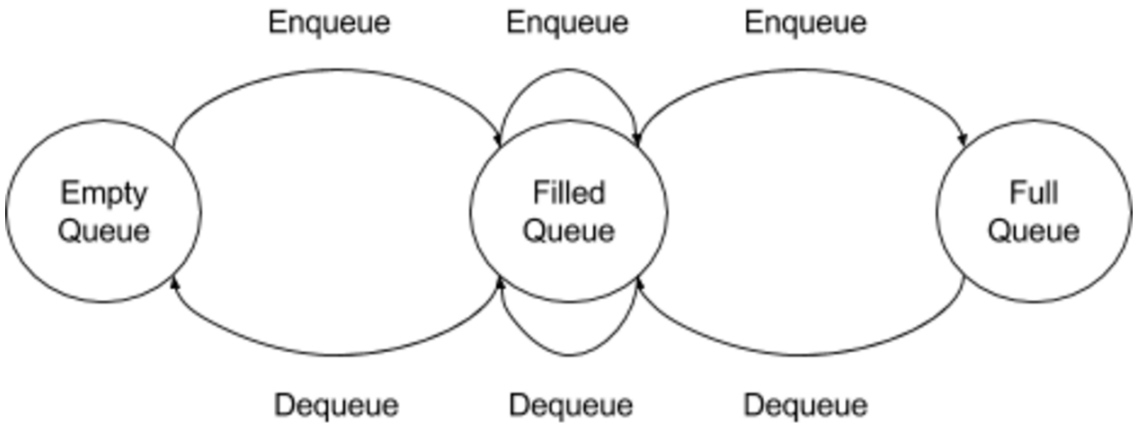

Figure 1: State transition system for a queue

In figure 1, the queuing system consists of three different states: empty, filled, and full with transitions occurring when objects enter or leave the queue. The transition from the filled state back to itself may seem odd because the graph appears to suggest that a system state can transition back to itself.

However, what the graph is actually communicating is that the queue is capable of holding many objects at once. If the cyclic transition from the filled state back to itself did not exist, we could infer that the queue would only be able to hold, at most, two objects. The first object to enter the queue would transition the system into the filled state, and a second object would transition the system into the full state. Obviously, this type of queue would be ineffective, therefore, the filled state must be able to accommodate many objects.

So what are some practical applications for state transition systems? While our example demonstrates a fairly trivial system, most modern systems are much more complex. Consider an operating system and how it needs to manage resource allocation for a large number of processes. A state transition system can help us determine when important system events such as resources are used, what process will be using resources, and how long they will be used for.

State transition systems can help us understand how parallel processing, multithreading, or distributed computing can affect the overall state of our system. Do we need to wait for another process to finish its work before continuing? At which junction of our system will we be bottlenecked?

In addition, state transition systems can help us identify deadlocks. A deadlock occurs when a process is waiting indefinitely for another to release a shared resource or complete its work. A state transition system can help you easily identify deadlocks since they occur if there is a condition that prevents a transition out of particular state.

Modern day software continues to become more complex. Their reliance on computing techniques such as multithreading makes it more difficult to manage system resources, or determine the state that your system is in. State transition systems helps alleviate this issues by modeling the behavior of a system, and gives you a better understanding of how a system will transition from one state to another.