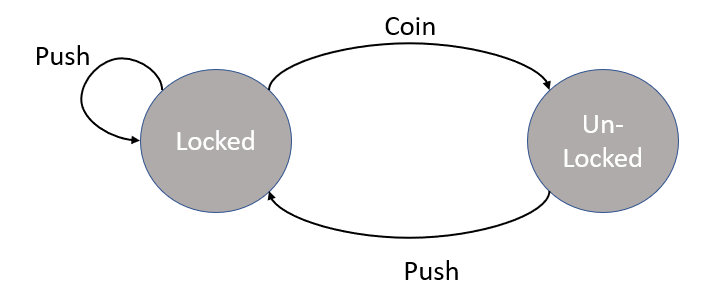
**What is a Finite-State Machine?**

Skipping the math, we can define a finite-state machine as an algorithm. This algorithm contains a finite number of states representing independent operations. Each state includes at least three attributes: (1) A set of operations that must be executed when the state is active. (2) A set of input stimuli. (3) Output transitions that will move from one state to another.

The most common example of a finite-state machine is the coin operated turnstile. A turnstile has two states: locked and un-locked. If the turnstile is in the locked state, it will remain in the locked state until a coin is used for the finite-state machine to transition to the unlocked state. Once in the unlocked state, pushing the turnstile will then move it back to the locked state. This logic approach is very systematic and coverts easily using the Xg Engine Library. Within Xg, each state is represented by a XgState object. A XgState object can contain any number of events that can be used to transition. The first one to test true will transfer the finite-state machine to another state. These events can include keyboard entries, mouse moments, and collision detection. Within the Xg Engine Library a finite-state machine is known as a framework. Any object can be given a framework allowing complete control over the objects behavior.