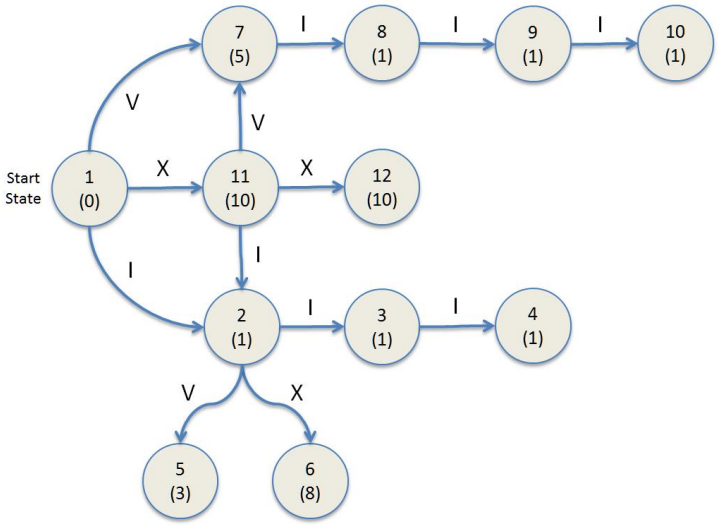
FSM-based Solution to Roman Numeral Conversion

Another interesting solution to the roman2 problem is based on Finite State Machines (FSM). FSM is a widely used system modeling methodology in engineering. Here we'll create an FSM that will process a string representing a roman number one character at a time from left to right.

An FSM consists of a set of states (nodes of a graph) and a set of state transitions (directed edges of a graph). The system is always in exactly one state. Based on the input, it can then move from the current state to the next state. There is exactly one start state. If there is no state transition corresponding to an input in the given state, then the input is illegal.

Consider the figure below representing the FSM that can process roman numbers from 1 to 20 (I to XX). The states are named by integers from 1 to 12 and the numbers in parentheses are the values of the states. As we go through the states as we receive the input characters one by one, we accumulate the value as we transition into a state. When there is no more input, the accumulated value is the final value of the roman number.



For example, if the input is "XIV", we start in state 1. "X" makes us move to state 11 and add 10 to the current value of 0. Then "I" takes us to state 2 and we add 1 to the value. Finally, "V" moves us to state 5 and we add 3 to get exactly 14 as expected. The tricky part is the last transition into state 5. Why is the value of that state 3 when we had a "V" input? That's because we already counted 1 for "I" when we moved to state 2 and "IV" means 4, so we need to add 3 to 1 to get 4.

The function roman2 on the class website has the FSM capable of handling roman numbers up to 399, so it has more states and many more state transitions than what we have in the figure here. But the idea is the same. The variable "states" contains the values of the states, while the index into states is the identifier of the state. The matrix "trans" contains the state transitions. Each row is one transition with the first column containing the current state, the second column having the input character and the third column storing the next state.

The actual MATLAB code of the roman2 function is generic. It only knows about states and transitions and how to execute the FSM based on the elements of the input vector (it does not even have to be char). It does not know anything about roman numbers.