- The property sees the start of its sequence. In this case q was TRUE. Nothing is printed and an assertion thread starts up to follow the sequence.
- The property doesn't see the start of its sequence because q is FALSE. In assertion-speak this is termed a vacuous success. A *vacuous success* means that it succeeded in *trying* to start an assertion thread, but the thread didn't start. The property continues trying to recognize the start of its sequence at the next clock tick.
- The assertion thread, having seen the start of its sequence earlier, didn't see one of the subsequent steps. In this case the assertion fails, the assertion thread stops, and the fail_Statement is executed. In this case it would print "Noooo!" and continue with the simulation.
- The assertion thread, having correctly seen an earlier part of its sequence, sees the next (but not last) step. It continues watching for future steps.
- The assertion thread, having correctly seen all but the last step in the sequence (sometimes called the *penultimate* step), sees the last step. The assertion "passes" and executes the pass_Statement, in this case printing "Yesssss!". At this point, the assertion threads stops as there is no more sequence to follow.

9.2.4 A State Transition Diagram View of a Sequence

Sequence abxxc above can be thought of as a state transition diagram as shown in Figure 9.2. As in any state transition diagram in logic design, the clock tick drives the change of state as per the inputs of the system. In this case, the inputs are the inputs to the sequence.

In state 1, if q is not seen, then it's a vacuous success and the sequence stays in state 1. If q is TRUE, then state 2 is entered. When in state 2, if r is TRUE then state 3 is entered. If r is FALSE in state 2, the assertion fails. Three clock ticks later when in state 5, if s is TRUE, then the assertion passes. If not, it fails.

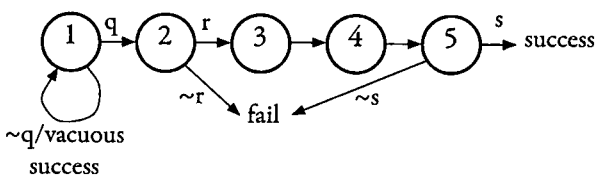

Figure 9.2 — State Transition Diagram for Sequence abxxc

As mentioned above, the assertion thread only checks for the specified variables (or expressions) to be TRUE. Thus, for instance, the assertion thread doesn't check for s to be FALSE in states 3 and 4.

An analogy for how SystemVerilog assertions and their sequences work is regular expressions; there is a one-to-one translation between regular expressions and state transition diagrams.

9.2.5 More Detail on the Previous Example

An example will show how many of these details come together in the execution of an assertion as it follows a simple waveform. The example waveform is still the one shown in Figure 9.1 where the sequence to check is (q ##1 r ##3 s).

A sequence and property written for this situation could be the following:

```
1  sequence s1 (q, r, s);
2    (q ##1 r ##3 s);
```