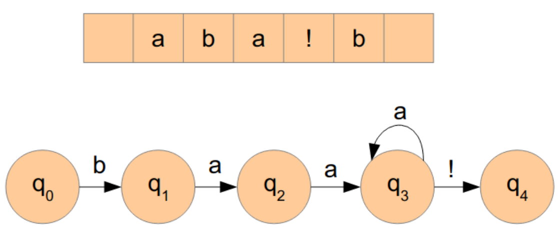
1 FSA and Regular Expressions

1.1 Finite State Automata

- A finite state automaton (FSA) is a directed graph with a finite number of nodes.
- A FSA is described by a 5 tuple: (states, alphabet, initial state, final state, transition)



	Input		
State	b	a	!
0	1	ø	ø
1	ø	2	ø
2	ø	3	ø
3	ø	3	4
4	ø	ø	ø

Figure 1.3: Example FSA representation

- A deterministic FSA is one whose behaviour is fully determined by the state it is in and the input
- A non-deterministic FSA has an element of stochasticity; perhaps two paths for the same input, or an ϵ (i.e. spontaneous) transition
- NDFSA's present challenges when determining whether a string should be accepted (by the language) as 'the wrong path' may be taken. A solution to this is to use a backtrack algorithm.
- Any NDFSA can be converted to a DFSA (See: parallel algorithm)
- An example of a DFSA suitable application is compiler scanning, and NDFSA is python RegEx

1.2 Regular Expressions

Regular expressions are a powerful tool for pattern matching. They are one way to define an FSA, and also to define a formal (specifically regular) language. Any regular expression can be defined as an NDFSA and hence a DFSA

(Note: A formal language is a set of strings that are composed entirely from a finite symbol set Σ)

1.3 Finite State Transducers

- A FST is a more general function than an FSA. It has two memory tapes, as opposed to FSA's one.
- It 'reads' one string and generates another.
- Each 'state' can hold any value from the alphabet, and each outward arc contains a 'state/input' pair

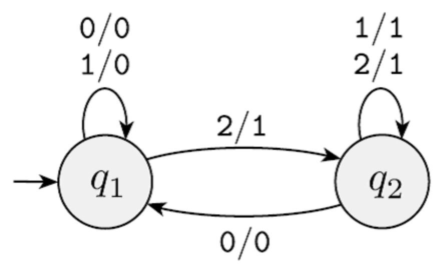


Figure ??: Example FST representation