Topic: 4.2.3 State-transition diagrams

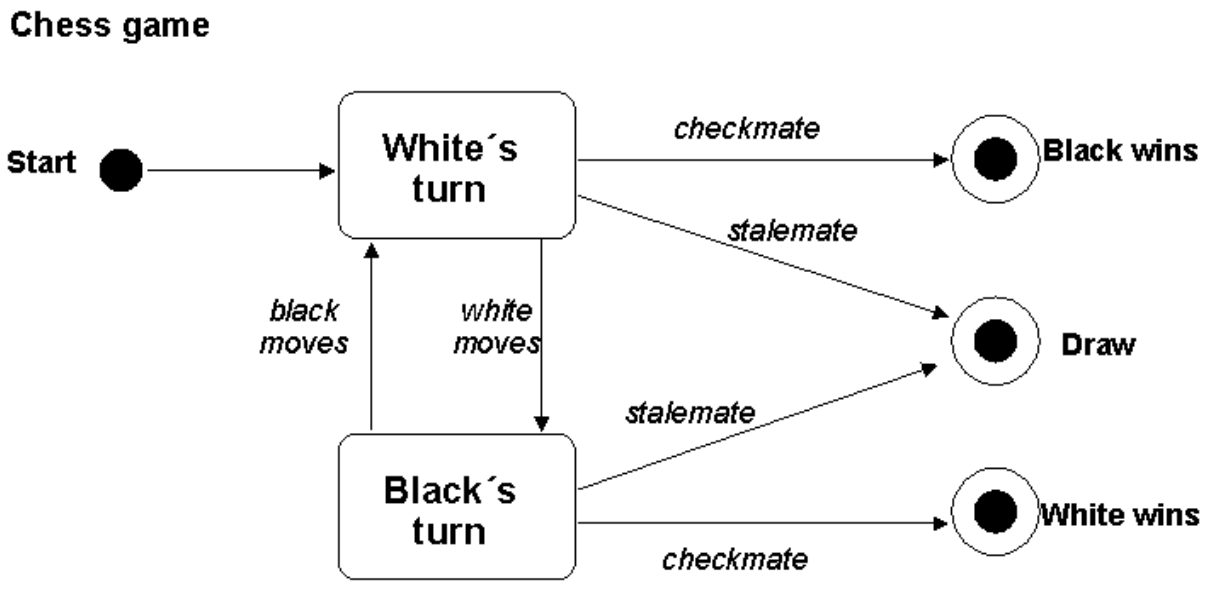
A finite state machine consists of states, inputs and outputs. The number of states is fixed; when an input is executed, the state is changed and an output is possibly produced. Finite state machines are widely used when designing computer programs, but also have their uses in other fields thanks to their ability to recognize sequences.

The finite state machine expressed visually is a **State transition diagram.** It is used to show all the states, inputs and outputs. Each state is represented with a circle, and each transition with an arrow. Transitions are labelled with an input that causes a transition and possibly an output those results from it. A double circle shows an accepting state. Not all finite state machines will have an accepting state and it is possible they could run forever.

States:

- A state is an observable mode of behavior of the system.
- At any time, a particular STD can only be in one state.
- You can describe a system's behavior with more than one state transition diagram.

Consider the following example:

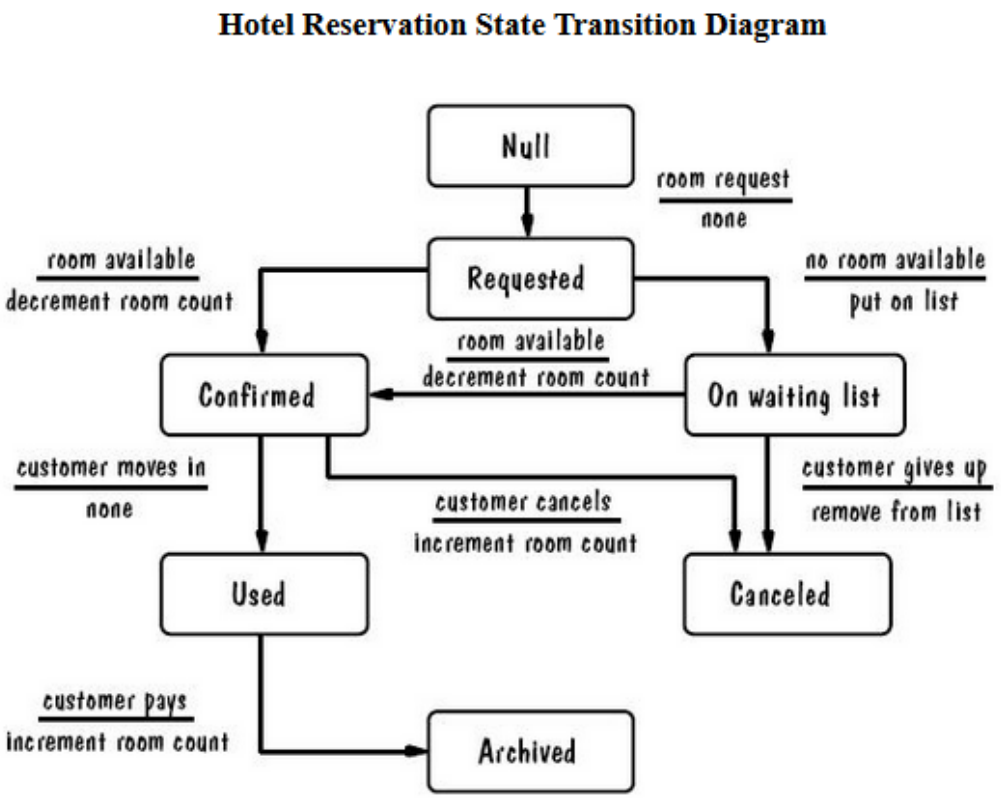


In the example above, there are only 2 possible states, **White's turn** and **Black's turn.** The STD can only be in one state at a time. According to the STD, the players keep taking turns until either a player wins, or the game ends up in a draw.

Topic: 4.2.3 State-transition diagrams

- The black circle on the left is the start state.
- The double circles on the right are the accepting states.
- The arrows are the transitions with descriptions written above the arrows.

Another example:



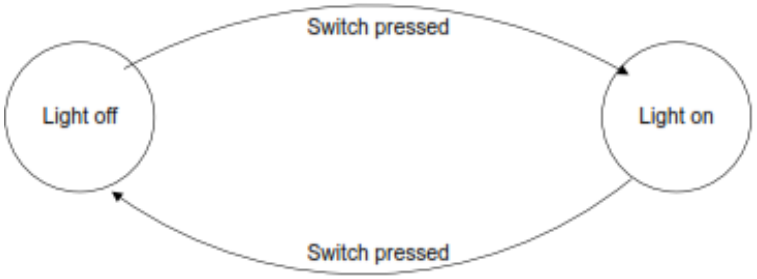
The diagram is fairly easy to understand, all of the boxes are states, and all of the arrows are transitions with description written above the line and the effect on the system written below the line.

Topic: 4.2.3 State-transition diagrams

A desk lamp has 2 states:

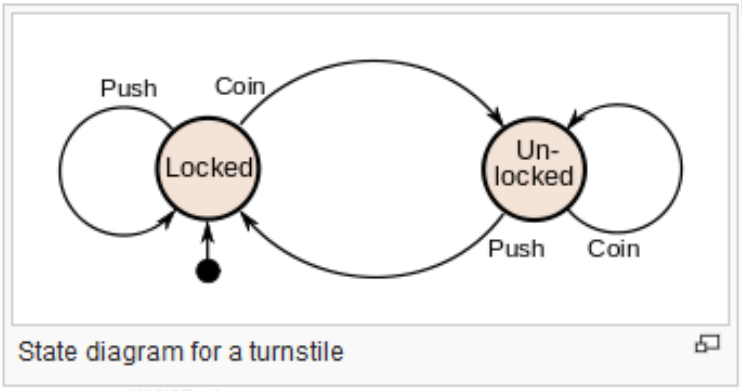
1. Light on
2. Light off

The input is to press the switch. This can be shown in a State transition diagram.



A turnstile can be modeled by a STD. A turnstile is used at amusement parks or subways to prevent passengers from going through without paying a fee.


A turnstile



State diagram for a turnstile

From the diagram above, you can see that if you keep pushing the turnstile when it is locked without putting a coin, it will remain locked. When you put a coin inside, it will unlock and you can push it and go through. If you insert more than one coin after the turnstile has been unlocked, it makes no difference. After the turnstile has been pushed, it goes back to the locked state and the process starts all over again.

The 2 states here are **Locked** and **Un-locked** and the arrows are the transitions with descriptions written.

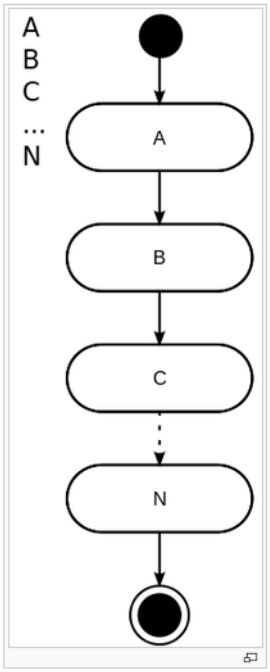
Topic: 4.2.3 State-transition diagrams

All state transition diagrams can have a table to make analyzing the behavior of the system easier. This table is called a **state transition table** showing for each state the new state and the output (action) resulting from each input.

For the turnstile example above, the table could be made as follows:

Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlock turnstile so customer can push through
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When customer has pushed through, lock turnstile

Just as STD's can be used to model the behavior of an object, it can also be used to document an algorithm, in other words it can be used to model the flow of operations of different algorithms similar to a flowchart.



Take the following code for example:

```
Console.writeline("This is line 1")
Console.writeline("This is line 2")
Console.writeline("This is line 3")
```

Just by looking at the code, you can tell that the compiler will simply execute instructions one after the other in a sequence. First, it will print “This is line 1” and so on… there is no alternate route the algorithm can take, so it is safe to model the above code in a state transition diagram as indicated. The model on the right is just a general diagram, but for the code above, there will be 3 states structured similar to the diagram on the right.

Topic: 4.2.3 State-transition diagrams

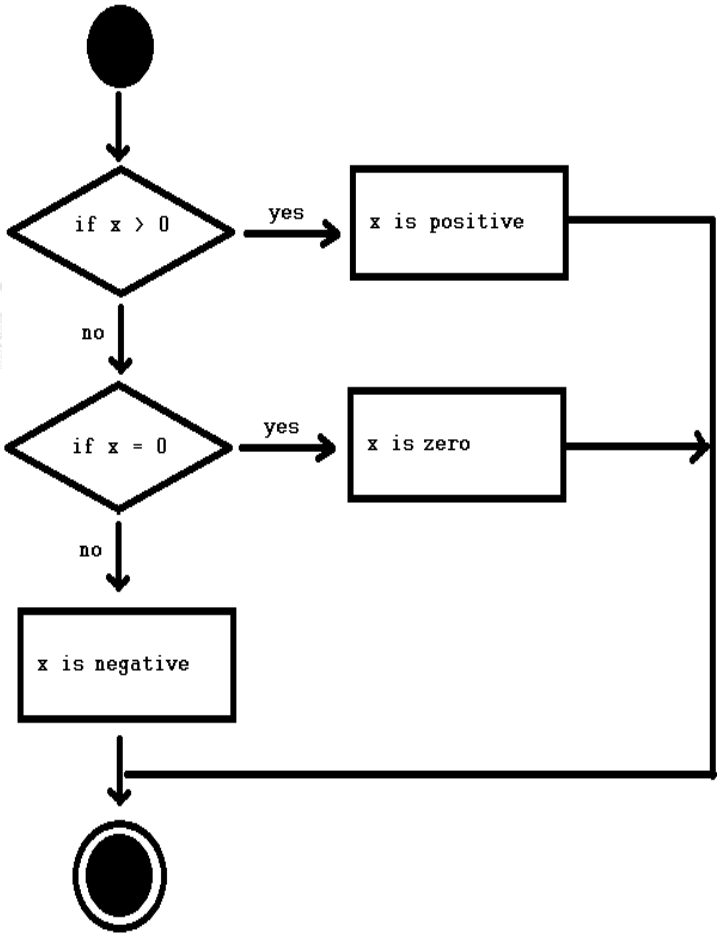
If you wish to assign a value to a variable such as:

```
Dim x, y, sum as integer
x = 5
y = 15
sum = x + y
```

This code is also executed in sequence by the compiler, so the STD for this code will be similar to the one above since there is only one path the algorithm can take.

In the scenario when the algorithm has 2 or more paths it can take, such as:

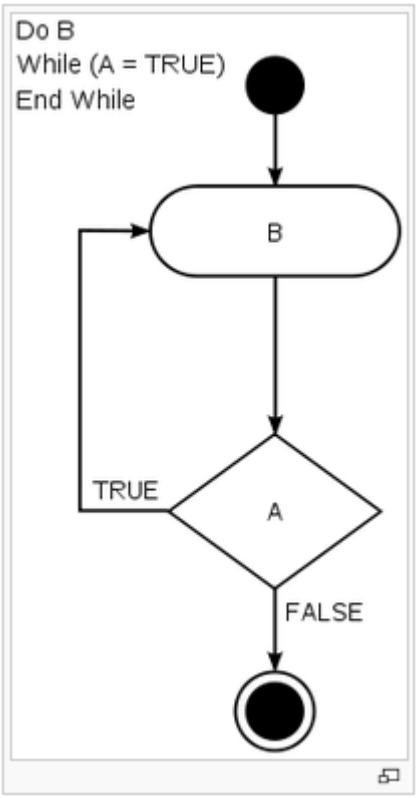
```
If x > 0 then
   Console.writeline("x is positive")
End If
If x = 0 then
   Console.writeline("x equals 0")
End If
If x < 0 then
   Console.writeline("x is negative")
End If
```

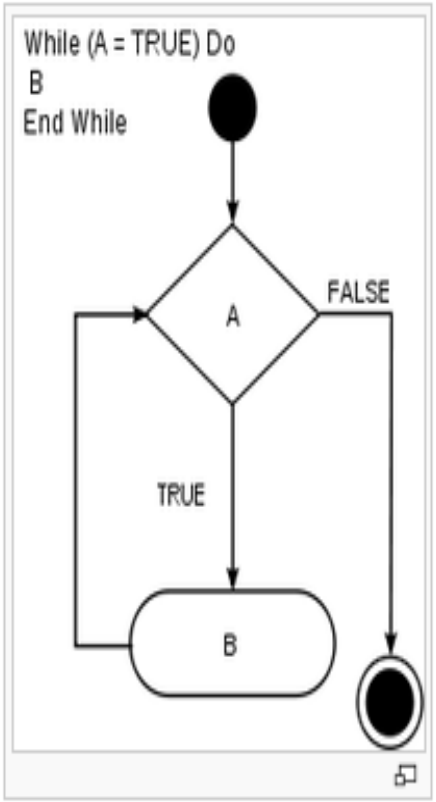


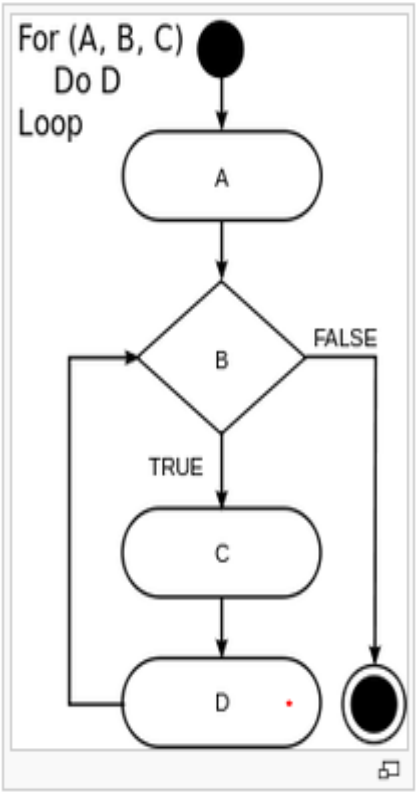
Topic: 4.2.3 State-transition diagrams

For repetition, a sequence of steps must be executed until a requirement is met.

Examples:







Try making a state transition diagram for the following code:

```
x = 0
y = 5
Do
   x = x + 1
Loop Until x = y
```