AQA Computer Science A-Level

4.4.2 Regular languages

Advanced Notes

<u>**Specification:**</u>

4.4.2.1 Finite state machines (FSMs) with and without output

Be able to draw and interpret simple state transition diagrams and state transition tables for FSMs with no output and with output (Mealy machines only).

4.4.2.2 Maths for regular expressions

Be familiar with the concept of a set and the following notations for specifying a set: A = {1, 2, 3, 4, 5 } or set comprehension: A = {x | x ∈ ℕ ∧ x ≥ 1 } where A is the set consisting of those objects x such that x ∈ ℕ and x ≥ 1 is true.

Know that the empty set, {}, is the set with no elements.

Know that an alternative symbol for the empty set is Ø.

A set is an unordered collection of values in which each value occurs at most once.

Several languages support set construction. In Python, for example, use of curly braces constructs a set: {1, 2, 3 }. | means such that. x ∈ ℕ means that x is a member of the set ℕ consisting of the natural numbers, ie {0, 1, 2, 3, 4, … }.

The symbol ∧ means AND. The term ∧ x > = 1 means AND x is greater than or equal to 1. In Python, {2 ∗ x for x in {1, 2, 3 }} constructs {2, 4, 6 }. This is said to be a set comprehension over the set {1, 2, 3 } .

Be familiar with the compact representation of a set, for example, the set {0n 1n | n ≥ 1}. This set contains all strings with an equal number of 0 s and 1s. For example, {0n 1n | n ≥ 1} = {01, 0011, 000111, 00001111, … }

Be familiar with the concept of:

- finite sets
- infinite sets
- countably infinite sets
- cardinality of a finite set
- Cartesian product of sets.

A finite set is one whose elements can be counted off by natural numbers up to a particular number, for example as: 1st element, 2nd element, …, 20th (and final) element. The set of natural numbers, ℕ and the set of real numbers, ℝ are examples of infinite sets.

A countably infinite set is one that can be counted off by the natural numbers. The set of real numbers is not countable.

The cardinality of a finite set is the number of elements in a set.

Cartesian product of two sets, X and Y, written X x Y and read 'X cross Y', is the set of all ordered pairs (a, b) where a is a member of A and b is a member of B.

Be familiar with the meaning of the term:

- subset. $\{0, 1, 2\} \subseteq \{0, 1, 2, 3\}$ where $\subseteq$ means subset of. $\subseteq$ includes both $\subset$ and =, for example $\{0, 1, 2, 3\} \subseteq \{0, 1, 2, 3\}$ is also true, because $\{0, 1, 2, 3\} = \{0, 1, 2, 3\}$.
- proper subset. $\{0, 1, 2\} \subset \mathbb{N}$ where $\subset$ means proper subset of, that is ℕ contains everything in {0, 1, 2 } but there is at least one element in ℕ that is not in {0, 1, 2 }.
- countable set. A countable set is a set with the same cardinality (number of elements) as some subset of natural numbers.

Be familiar with the set operations:

- membership
- union
- intersection
- difference.

The set difference A\B (or alternatively A-B) is defined by $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$.

4.4.2.3 Regular expressions

Know that a regular expression is simply a way of describing a set and that regular expressions allow particular types of languages to be described in a convenient shorthand notation. For example, the regular expression a(a|b)* generates the set of strings {a, aa, ab, aaa, aab, aba, …}.

Be able to form and use simple regular expressions for string manipulation and matching. Students should be familiar with the metacharacters:

- * (0 or more repetitions)
- + (1 or more repetitions)
- ? (0 or 1 repetitions, ie optional)
- | (alternation, ie or)
- ( ) to group regular expressions. Any other metacharacters used in an exam question will be explained as part of the question.

Be able to describe the relationship between regular expressions and FSMs. Regular expressions and FSMs are equivalent ways of defining a regular language.

Be able to write a regular expression to recognise the same language as a given FSM and vice versa. A student's ability to write very simple regular expressions and FSMs will be assessed.

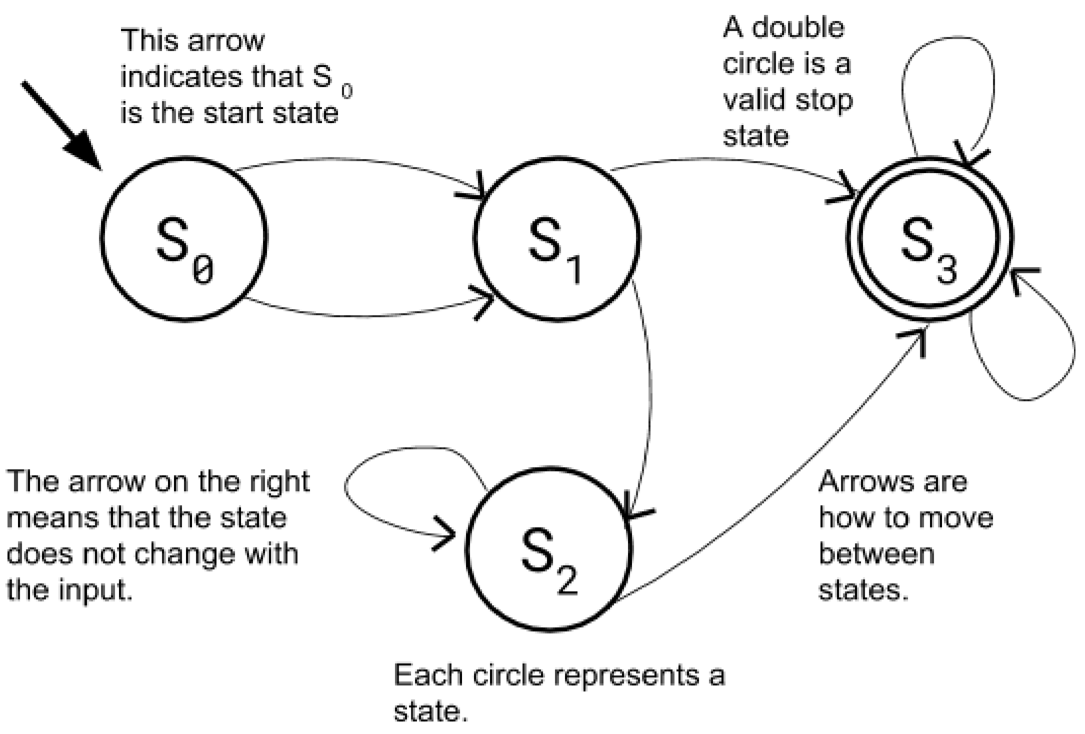
4.4.2.4 Regular language

Know that a language is called regular if it can be represented by a regular expression. Also, a regular language is any language that a FSM will accept.

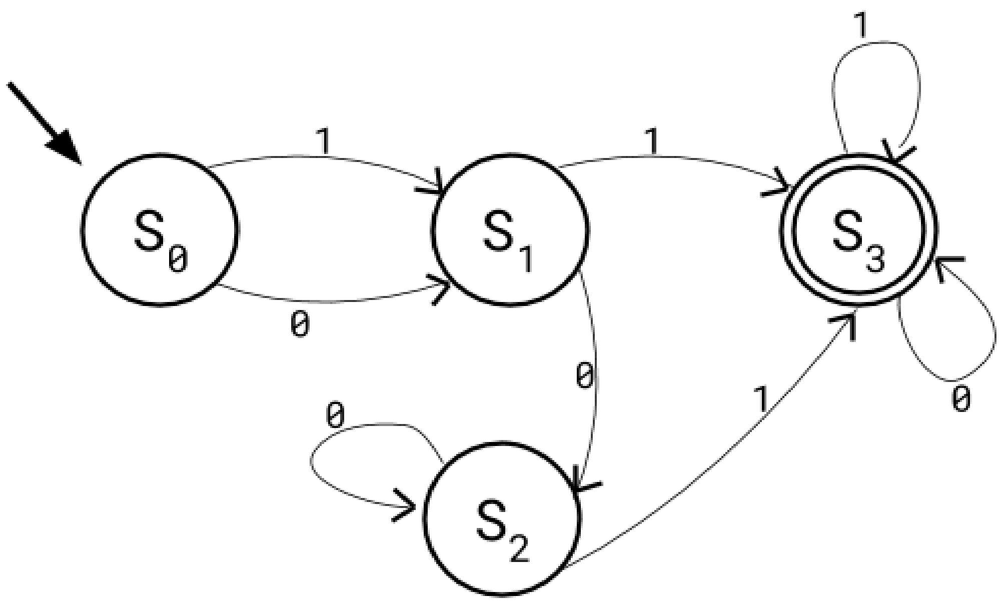
Finite State Machines

Finite state machines can have an output.

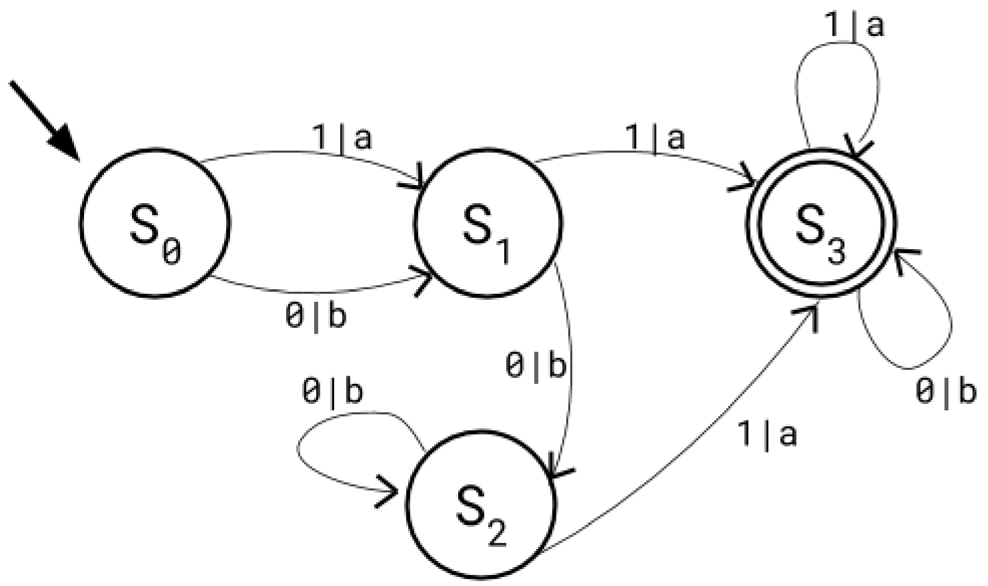
Here is a basic finite state machine.


This arrow indicates that S 0 is the start state
A double circle is a valid stop state
S0
S1
S3
S2
The arrow on the right means that the state does not change with the input.
Arrows are how to move between states.
Each circle represents a state.

Let's add some inputs



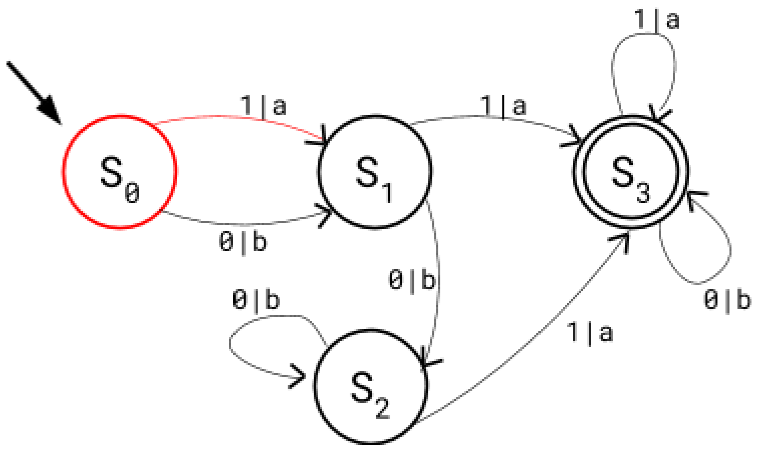
If a 1 is inputted at S0, the state changes to S1. If a 0 is inputted at S1, the state changes to S2 and so forth.
We could also add some outputs. These are separated from the inputs with a |.



An example input might be 1000101.

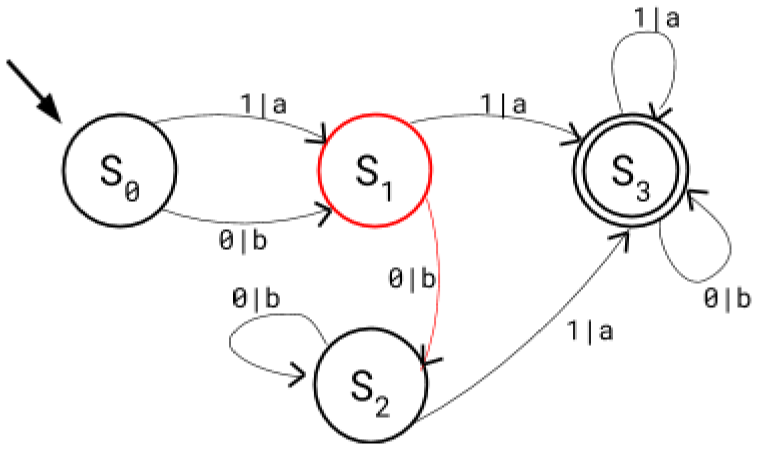
The start state (indicated by the arrow) is S_0, and our first input is a 1.

State	S_0						
Input	1	0	0	0	1	0	1
Output							



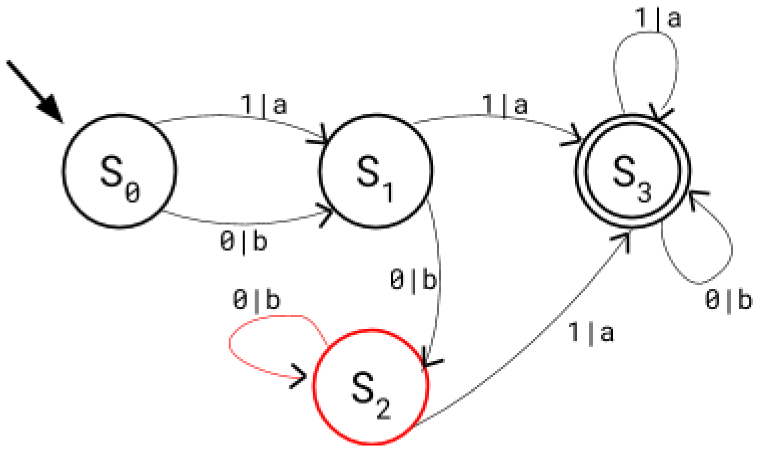
An “a” is outputted, and the state changes to S_1.The next input is 0.

State	S_0	S_1					
Input	1	0	0	0	1	0	1
Output	a						



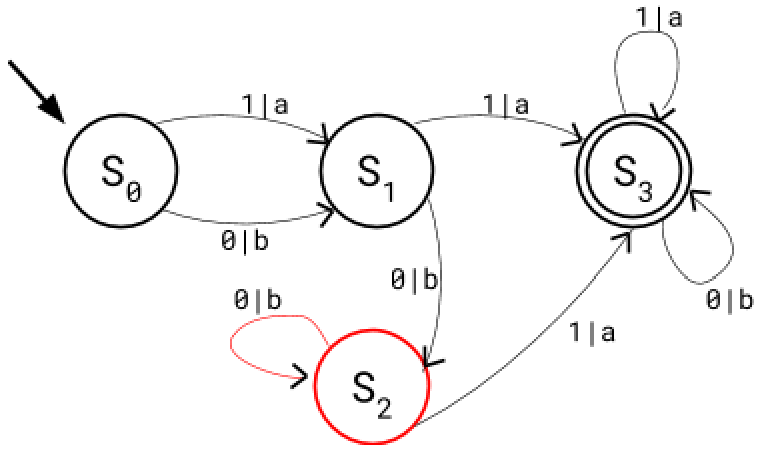
A “b” is outputted. The state changes from S1 to S2. The next input is 0.

State	S_0	S_1	S_2				
Input	1	0	0	0	1	0	1
Output	a	b					



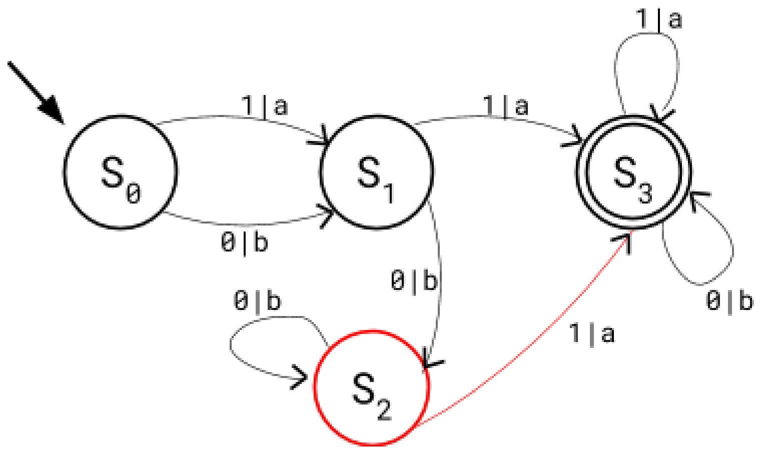
A “b” is outputted. The state stays as S2. The next input is 0.

State	S_0	S_1	S_2	S_2			
Input	1	0	0	0	1	0	1
Output	a	b	b				



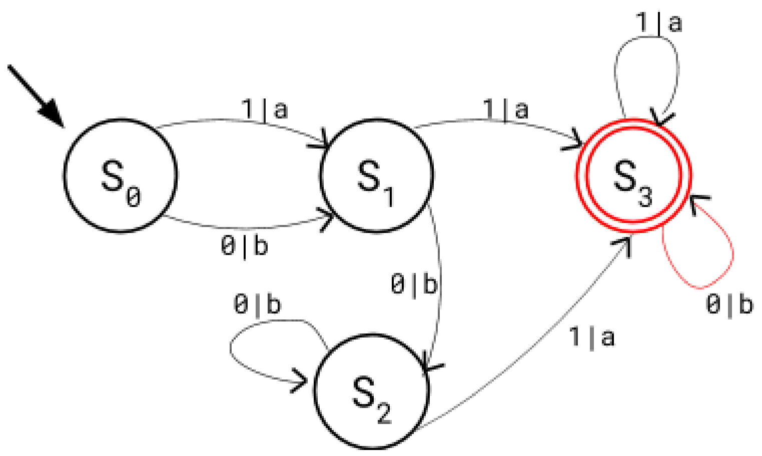
A “b” is outputted. The state stays as S2. The next input is 1.

State	S_0	S_1	S_2	S_2	S_2		
Input	1	0	0	0	1	0	1
Output	a	b	b	b			



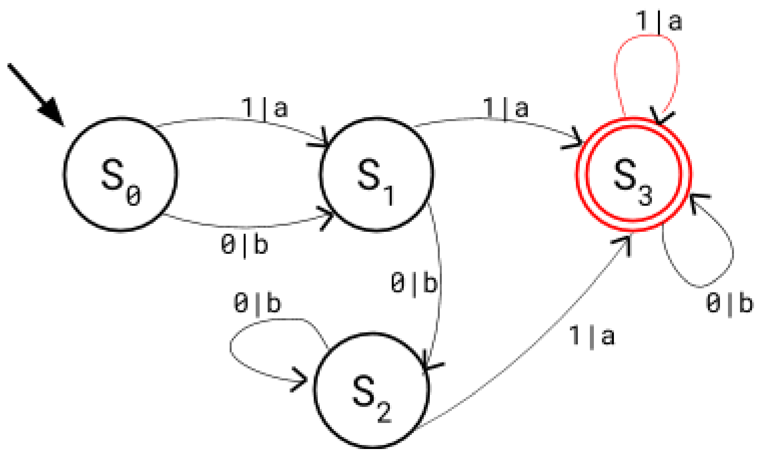
An “a” is outputted. The state changes from S2 to S3. The next input is 0.

State	S_0	S_1	S_2	S_2	S_2	S_3	
Input	1	0	0	0	1	0	1
Output	a	b	b	b	a		



A “b” is outputted. The state stays as S3. The next input is 1.

State	S_0	S_1	S_2	S_2	S_2	S_3	S_3
Input	1	0	0	0	1	0	1
Output	a	b	b	b	a	b	



An “a” is outputted. The state stays as S3. There are no more inputs.

State	S_0	S_1	S_2	S_2	S_2	S_3	S_3
Input	1	0	0	0	1	0	1
Output	a	b	b	b	a	b	a

The input 1000101 has landed on a valid stop state and produced the output “abbbaba”.

Sets

A set is an abstract data type which contains unordered unique values. Many languages support set construction, including Python. Sets can contain other sets. Below are examples of common set notation.

Set Example 1:
F is a set of farm animals. This is the notation for the set.

F: {“Pig”, “Goat”, “Cow”, “Sheep”}

Set Example 2:
H is a set of heights of 10 year old children. This is the notation for the set.

H: {143.1, 142.8, 145.0, 143.5}

Set Example 3:
A is the set of school subjects Charlie has on Monday.
A: {“Eng”, “Mat”, “MFL”}
B is the set of school subjects Charlie has on Wednesday.
B: {“Geo”, “Art”, “His”}
C is the set of school subjects Charlie has on Friday.
C: {“D.T”, “P.E.”, “Sci”}

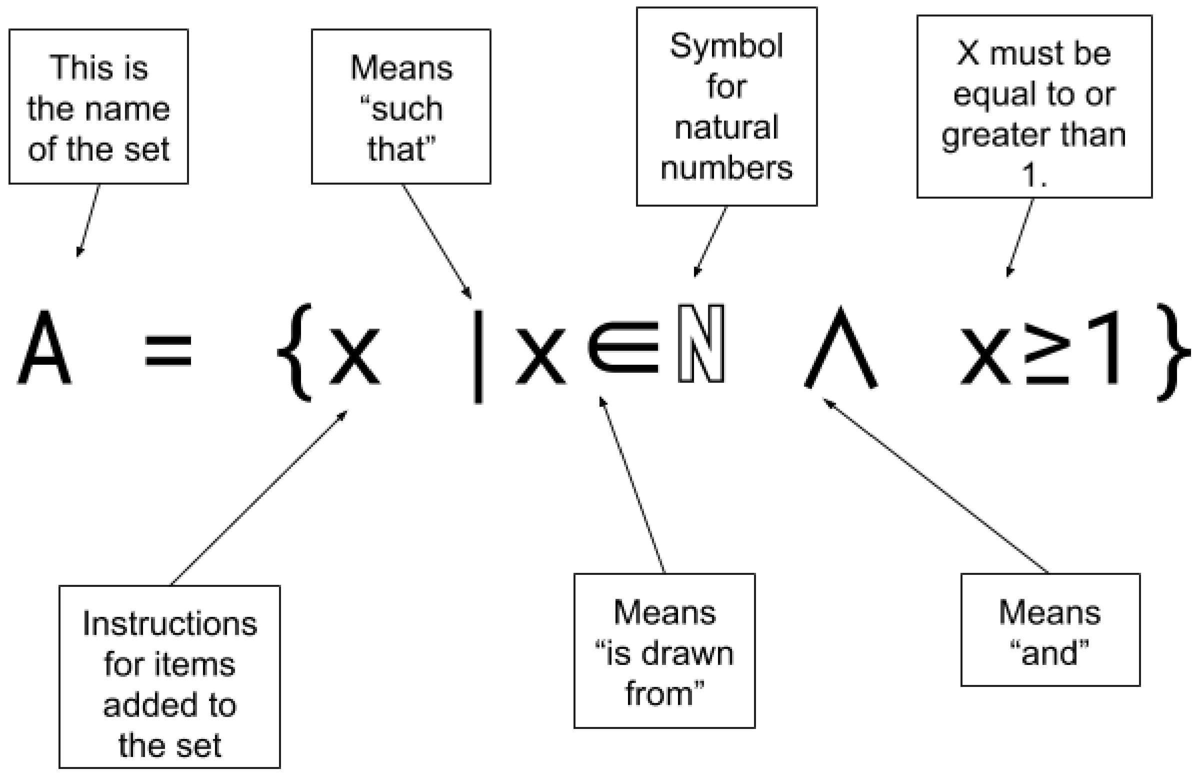
D is the set of the subjects Charlie has on Monday, Wednesday and Friday.
D: {(“Eng”, “Mat”, “MFL”), (“Geo”, “Art”, “His”), (“D.T”, “P.E.”, “Sci”)}

Python Set Example 1:
Using curly brackets in Python constructs a set. E.g. to construct a set containing 1, 2 and 3, the following code would be implemented:

```
{1, 2, 3}
```

Set Comprehension

Set comprehension is a different way of creating a set; rather than specifying all of the items individually, we can select what we want from a more general set. Below is the set comprehension for all the positive integers (not including 0).


This is the name of the set
Means "such that"
Symbol for natural numbers
X must be equal to or greater than 1.
$A = \{x \mid x \in \mathbb{N} \wedge x \geq 1\}$
Instructions for items added to the set
Means "is drawn from"
Means "and"

<u>Set Comprehension Example 1 - Understanding:</u>

What set would the following code produce?

$$S=\{x*3 \mid x \in \mathbb{Z} \wedge x \geq -3 \wedge x<3\}$$

The first thing to note is what general set the numbers are being drawn from.

$$S=\{x*3 \mid x \in \mathbb{Z} \wedge x \geq -3 \wedge x<3\}$$

The values are being drawn from Z, which stands for the set of integers. This includes all the positive and negative whole numbers and zero. Next, we look at the first condition.

$$S=\{x*3 \mid x \in \mathbb{Z} \wedge x \geq -3 \wedge x<3\}$$

We are only taking integers from Z which are greater than or equal to -3. In other words, we are taking -3, -2, -1, 0, 1, 2, 3, 4… to infinity. Now we look at the second condition.

$S=\{x*3 | x\in\mathbb{Z} \land x\geq-3 \land x<3\}$

This limitation means we are only taking integers below 3. By applying both of the conditions, we can say we are only looking at the integers between -3 and 2; they are -3, -2, -1, 0, 1 and 2. Finally, we look at the instructions for these numbers.

$S=\{x*3 | x\in\mathbb{Z} \land x\geq-3 \land x<3\}$

Each number specified should now be multiplied by 3, before being added to the set. This will be the result.

$S = \{-9, -6, -3, 0, 3, 6\}$

This is said to be a set comprehension over the set {-3, -2, -1, 0, 1, 2}.

Set Comprehension Example 1 - Creating:

How do we create a set of the first 10 square numbers (0 inclusive)?

S={Instruction|x∈**DefininedSet** ∧ x satisfies condition}

Firstly, we should investigate which set X should be drawn from. Square numbers rely on whole numbers, so we should either use integers or natural numbers. Technically either of these can be used, but it is better programming practise to use the most specific set. Hence, the values will be drawn from the natural numbers.

S={Instruction|x∈ℕ ∧ x satisfies condition}

We only want to take the first 10 square numbers (0 - 9); therefore x must be less than or equal to 9. X should also be equal to or greater than 0, but this is a constraint given by using natural numbers.

S={Instruction|x∈ℕ ∧ 9≥x}

Lastly, we add the instruction. All the values of x, from 0 to 9, should be multiplied by themselves. Hence, the instruction should square the values. This is the final set comprehension.

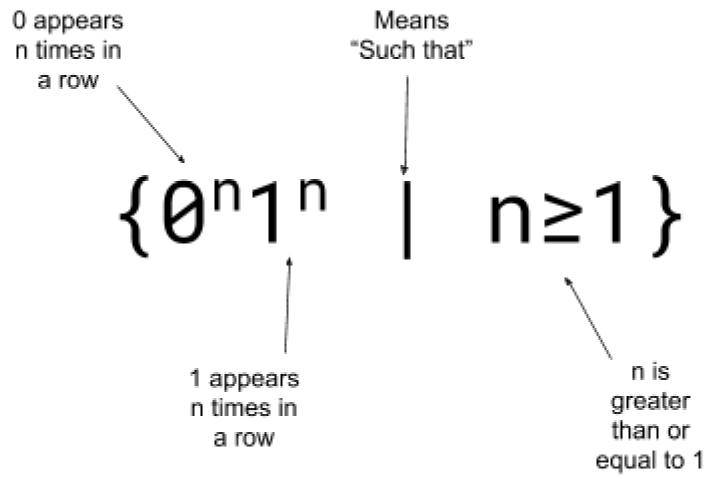
$$S=\{x*x \mid x \in \mathbb{N} \land 9 \geq x\}$$

Empty Sets

Sets can contain 0 elements. Such sets are referred to as empty sets. The symbols for empty sets are {} and Ø.

Compact Set Representation

Compact set representation is a space-efficient way of describing sets. This can include using shorthand ways of describing multiple instances of a number. The logic from set comprehension can be applied to compact set representation. The following example has been taken from the specification.


0 appears n times in a row
Means "Such that"
{0ⁿ1ⁿ | n≥1}
1 appears n times in a row
n is greater than or equal to 1

This set contains all strings with an equal number of 0 s and 1s.
$\{0^n 1^n \mid n \geq 1\} = \{01, 0011, 000111, 00001111, \dots\}$
There is no need to specify where n is drawn from - n must always be positive and whole (natural numbers).

Set Key Words

Here are some common types of sets.

Finite sets:
Finite sets contain a finite number of items, i.e. the values can be paired with consecutive natural numbers up to a particular number. The cardinality of a finite set refers to the number of elements in a set.

Finite Sets Example:
Here is a set of numbers.

```
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

If we take the natural numbers from 1 to 10, we can pair up each number from the set with a natural number.

```
{(1,2),(2,4),(3,6),(4,8),(5,10),(6,12),(7,14),(8,16),(9,18),(10,
20)}
```

Therefore this set is finite. It has a cardinality of 10.

Infinite Sets:
Infinite sets are the opposite of finite sets, in that they contain an infinite number of items, e.g. the set of integers. Infinite sets can be divided in two: countable and non -countable sets.

Countably Infinite Sets:
The items in a countably infinite set can be counted off by the natural numbers. Countably infinite sets include the integers and the natural numbers.

Countably Infinite Sets Example:
The set of numbers evenly divisible by three is infinite.

{3, 6, 9, 12, 15, ...}

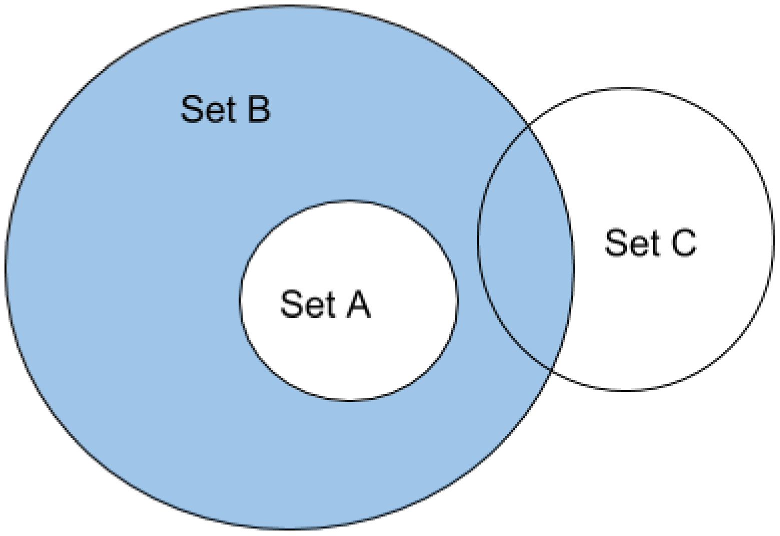
Each element can be paired with a natural number.

```
 1   2   3    4     5     ...
{3 , 6 , 9 , 12 , 15 , ...}
```

Therefore it is countably infinite. Instinctively, you may think that there are three times as many natural numbers than numbers evenly divisible by three. This misconception can cause problems in exams - any infinity which can be counted off by natural numbers are the same size as one another. So there are the same amount of multiples of three as there are of one.

Non-Countable Sets:
Non-countable sets contain a larger infinity of numbers - they cannot be paired with natural numbers. Non-countable sets include the real numbers.

Subset
Set A is a subset of Set B if it only contains items from Set B. The symbol for a subset is $\subseteq$. So if A is a subset of B, the notation would be A $\subseteq$ B, e.g. {2,4,6} $\subseteq$ {1,2,3,4,5,6}.



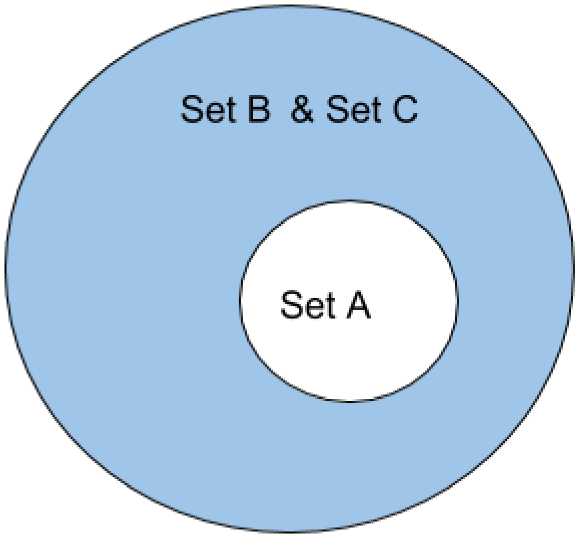
Set A is a subset of Set B. Set C is **not** a subset of B.

If set A was the same as set B, they would be subsets of each other e.g. {A,B,C} $\subseteq$ {A,B,C}.

Proper Subsets
Set A is a proper subset of set B if it only contains items from set B, but not all of them. Set A cannot be equal to set B to be a proper subset.

Note

Intuitively, you may think that Set A must have less items than Set B. However, set A may still have the same number of items as set B. The even integers is a proper subset of the integers, but both sets contain an infinite number of items.



Set A is a proper subset of Set B and of Set C. Set C is a subset of B, and Set B is a subset of C. Set C is **not** a proper subset of B.

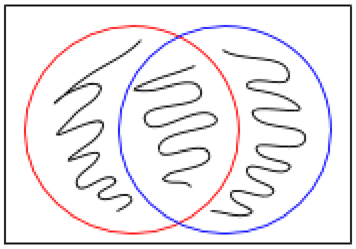
Countable Sets
Countable sets have the same cardinality as some subset of the natural numbers. Countable sets include finite sets and countably infinite sets.

Membership
If an item is in a set, then ∈ is used to denote this. E.g 3 ∈ R.
If an item is not contained in a set, then ∉ can be used to show this. E.g. -3.2 ∉ N

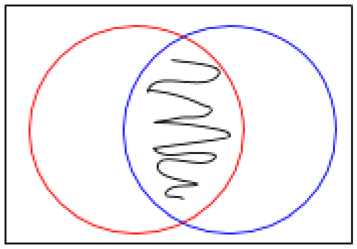
Set Operations

New sets can be constructed from other sets by using the following set operations.

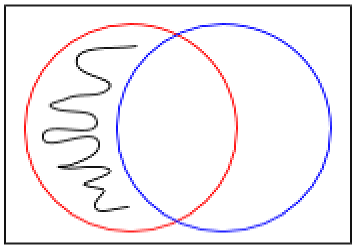
Union



Intersection

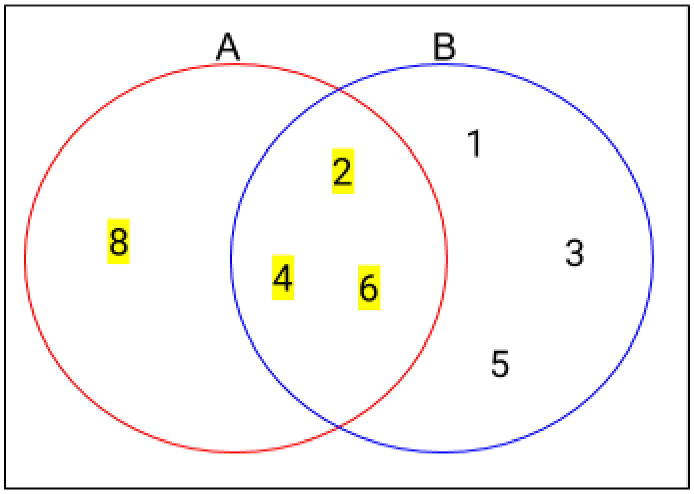


Difference



Union
A new set can be created through the union of two other sets. The symbol signifying a union is ∪. The values from each set are taken and added to the new set. Where a value appears in each set, it will only appear once in the new set.

Union Example:
Set A: {2,4,6,8}
Set B: {1,2,3,4,5,6}

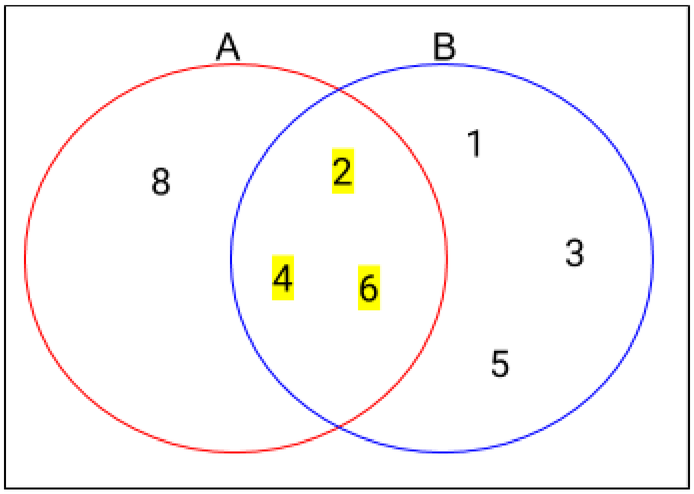


Set A ∪ Set B
>> {1,2,3,4,5,6,8}

Set A ∪ Set B is the same as Set B ∪ Set A

Intersection
If a new set is constructed from the intersection of set A and set B, then it will only include the values in both sets. The symbol for intersection is ∩.

Intersection Example
Set A: {2,4,6,8}
Set B: {1,2,3,4,5,6}



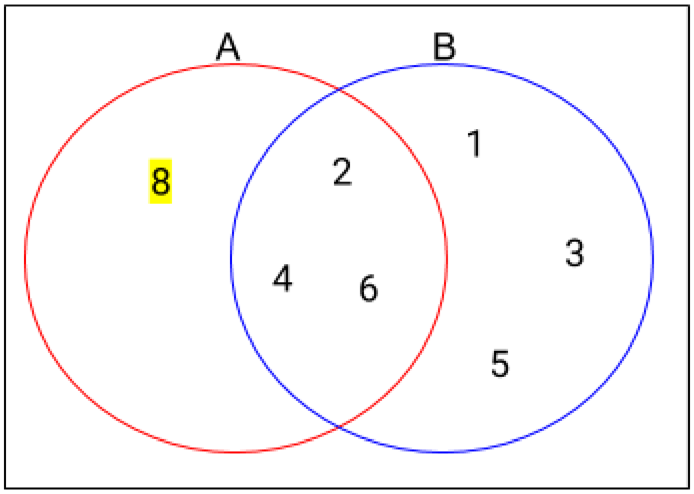
Set A ∩ Set B
>> {2,4,6}

Set A ∩ Set B is the same as Set B ∩ Set A

Difference
A set created by set difference will only contain items exclusive to one set. The set difference symbol is \ or -.

A\B = {x : x ∈ A and x ∉ B}.

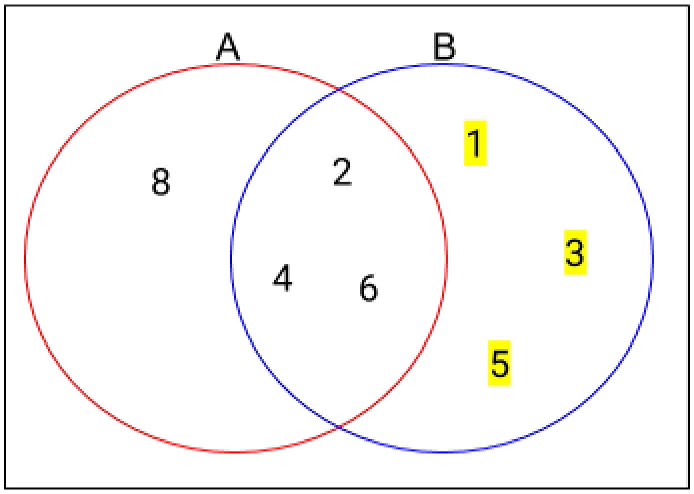
Difference Example a
Set A: {2,4,6,8}
Set B: {1,2,3,4,5,6}



A\B
>> {8}

A - B
>> {8}

Difference Example b
Set A: {2,4,6,8}
Set B: {1,2,3,4,5,6}



B\A
>> {1,3,5}

B - A
>> {1,3,5}

Regular Expressions

Regular expressions are shorthand ways of describing sets. There are several metacharacters used in regular expressions. The ones below are the only ones you have to know. In another metacharacter appears in an exam it will be explained in the question.

Metacharacter	Description	Example
*	0 or more repetitions	The set described by ab* is {a, ab, abb, abbb …}
+	1 or more repetitions	The set described by c*d is {cd, ccd, cccd…}
?	Previous character optional	The set described by Colou?r is {Colour, Color}
|	Alternative/ or	The set described by e|f is {e, f}
()	Used to group regular expressions	The set described by (ab)|(cd)e is {abe, cde}

<u>Regular Expressions Example 1</u>
What set does the notation the following notation describe?

a+(bc)*

First we need to know what letters the symbols are referring to. The symbols always refer to the letter/brackets immediately preceding it. Let's look at the plus symbol.

a+(bc)*

Before plus is a. The plus means one or more repetitions; the set denoted by a+ is {a,aa,aaa…}. Now we look at the second symbol.

a+(bc)*

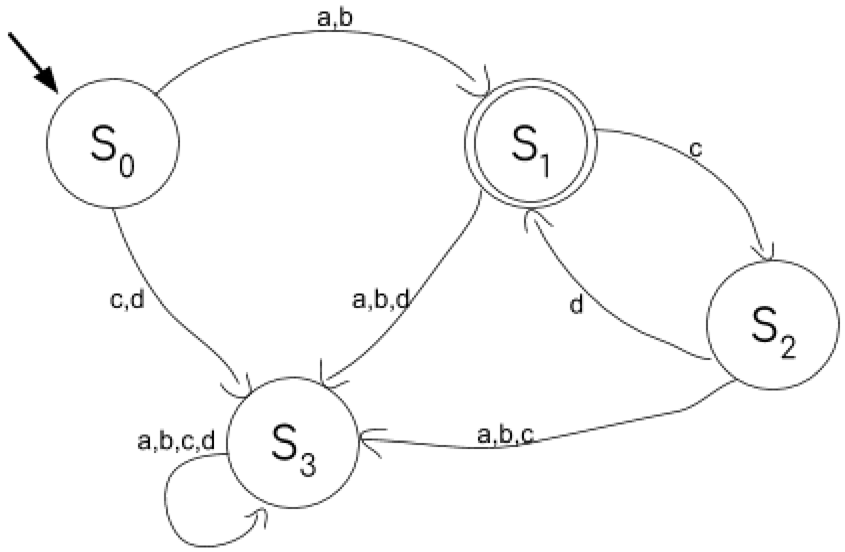
Immediately before the asterisk is a bracket, so the asterisk is referring to bc. The asterisk means that there should be 0 or more repetitions of bc. The set (bc)* is { ,bc,bcbc,bcbcbc}.

Here are some items contained in the set a+(bc)*
a
aa
aaa
abc
abcbcbc
aaaaaaaaabcbcbcbcbcbcbcbcbcbcbcbc

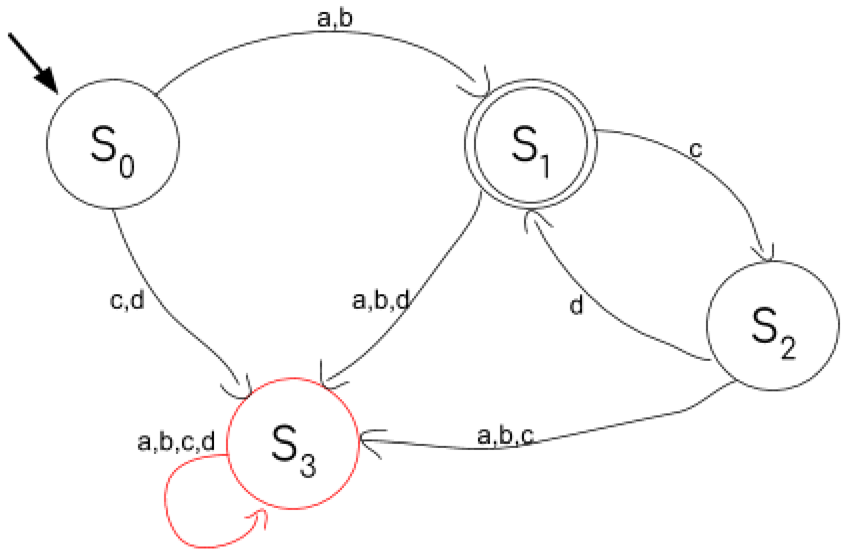
Regular Expression and FSMs

For each set denoted by a regular expression, there is an equivalent FSM.

Regular Expression and FSMs Example 1:
An FSM recognises the alphabet a, b, c and d. What is the equivalent regular expression to the following FSM?



The first thing to notice is S_3.

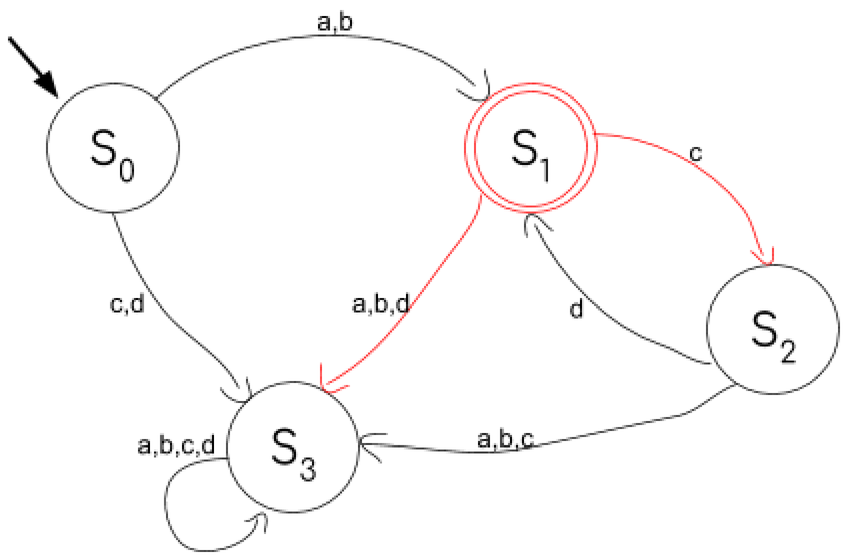


It is not a stop state, but once at S_3 there is no way to leave. This means the input should be rejected.

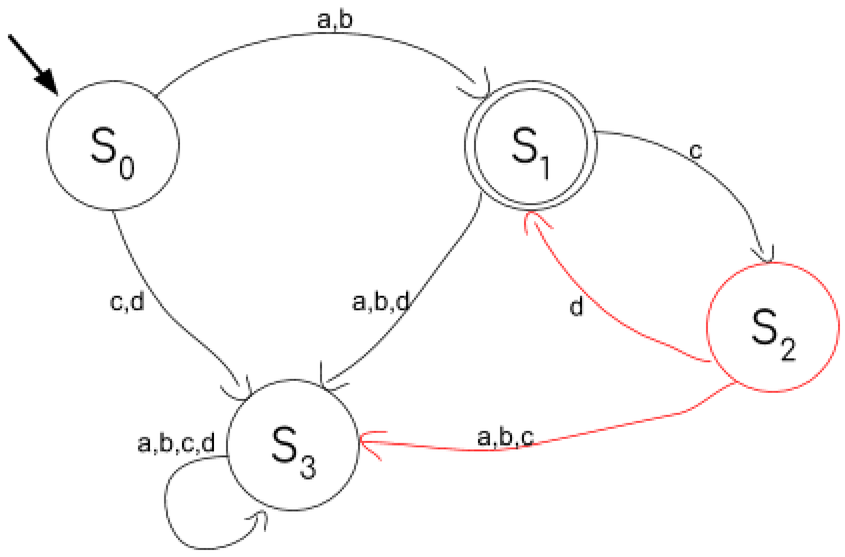
The initial state is S_0. To avoid S_3, an a or b must be entered. The notation for a or b is the following:

$$a \mid b$$

This is the first part of the regular expression. S1 is a valid stop state, so a must be an acceptable input and b must be an acceptable input. Now we should look at S_1.



To avoid going to the reject state, either a c must be entered or nothing should be entered. By entering a c, you will be sent to S_2.



At S_2, you must enter d to avoid the reject state. In other words, after entering a or b, you can enter cd as many times as you like, with a minimum of 0 times. This can be written as:

(cd)*

The full regular expression is below.

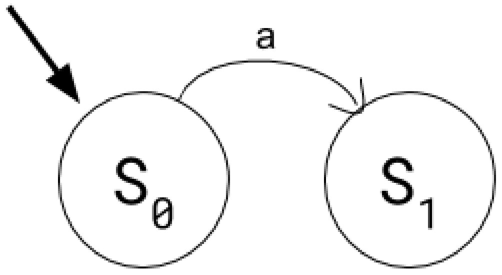
a|b(cd)*

Regular Expression and FSMs Example 2:
How do you turn the following regular expression into an FSM?

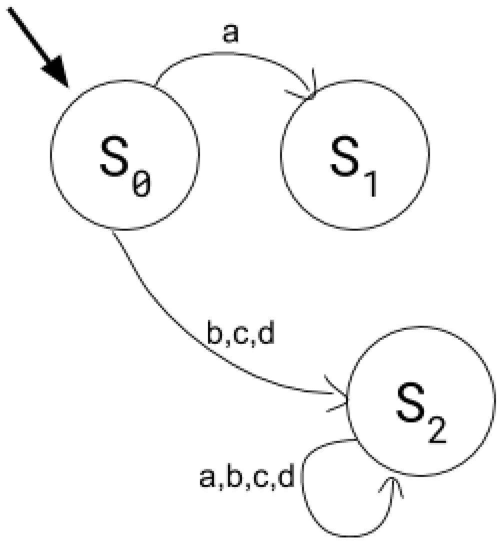
$$(abc)+b*d?$$

The finite state machine only needs to recognise an alphabet of a b c and d. The first symbol is the plus which refers to abc. This means that abc must appear at least once.

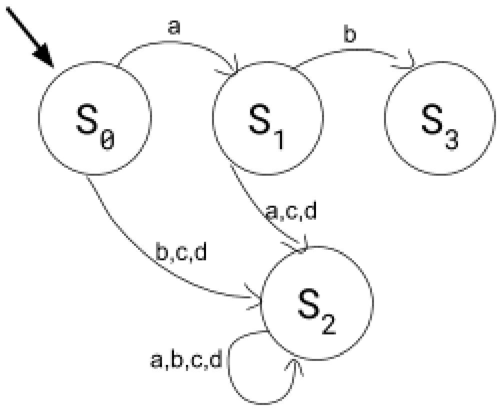
The first letter inputted needs to be a.



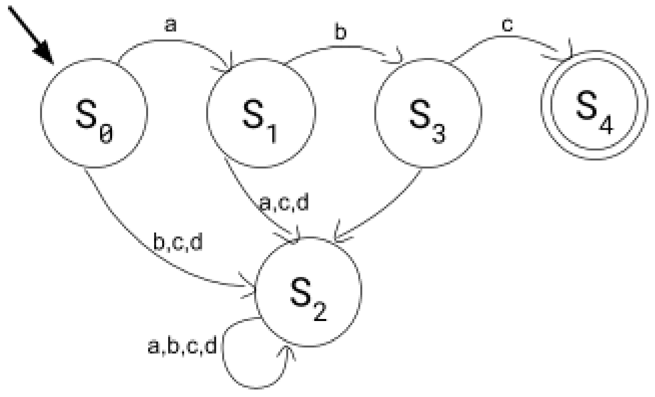
If an a isn't entered, the input should be rejected. S_2 can be the reject state.



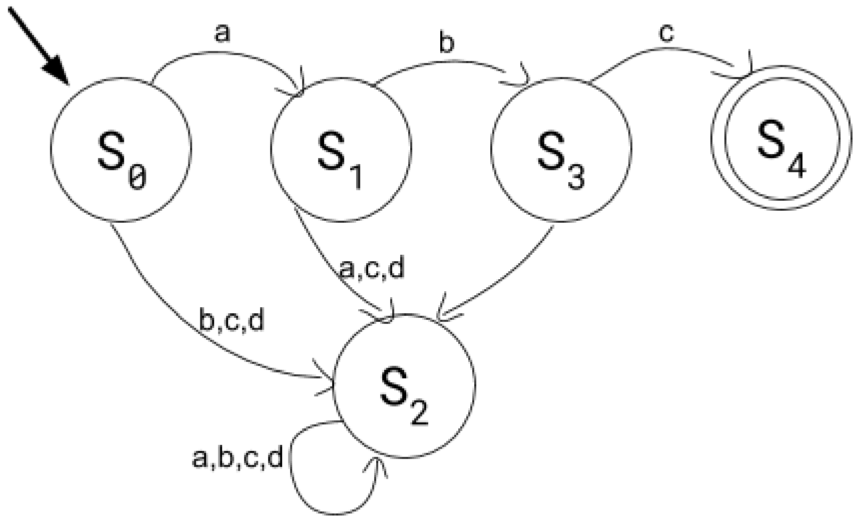
After a, a b must be entered, anything else should be rejected.



After b, c should be entered and anything else should be rejected. As the * and ? are optional, the next node is a valid stop state.



From S_4 you need the option to repeat abc as many times as you want. This can be added like this:


a
b
c
S_0
S_1
S_3
S_4
a,c,d
b,c,d
S_2
a,b,c,d