



Computational Linguistics

Question hour

Pierre Lison

Language Technology Lab
DFKI GmbH, Saarbrücken

<http://talkingrobots.dfki.de>

Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence





- We'll review the questions of the Probeklausur, one by one
- Note that I'm not a specialist in all of these topics, but I'll do my best to re-explain the main concepts
- If you have any questions, please don't hesitate!
 - Thursday, it will be too late :-)

A. Finite-state algorithms

1. Draw the automaton A as specified below. Which language does this automaton accept?
2. Construct an equivalent deterministic automaton A' by subset construction.

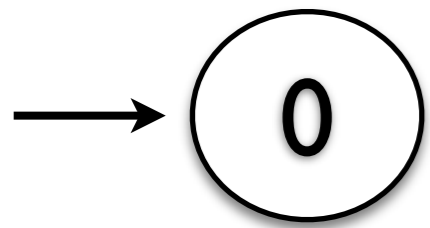
$A = \langle \{0,1,2,3,4,5,6\}, \{a,b,c,\epsilon\}, \delta, 0, \{6\} \rangle$, with initial state 0 and the set of end states $F=\{6\}$.

δ	a	b	c	ϵ
0				1,2
1		3		
2	4			
3				1,5
4				2,5
5			6	
6				5

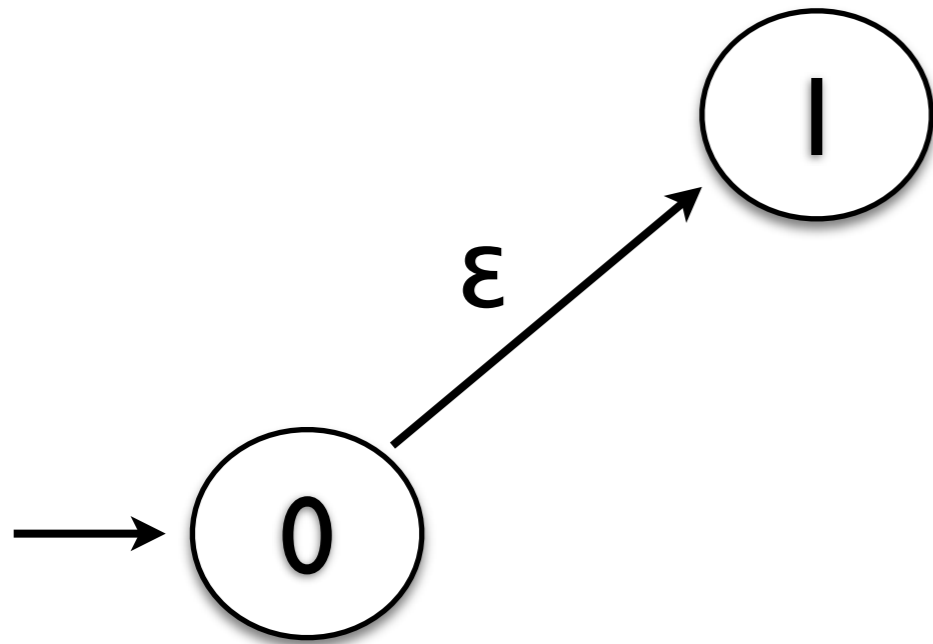
Q1.1: drawing the FSA



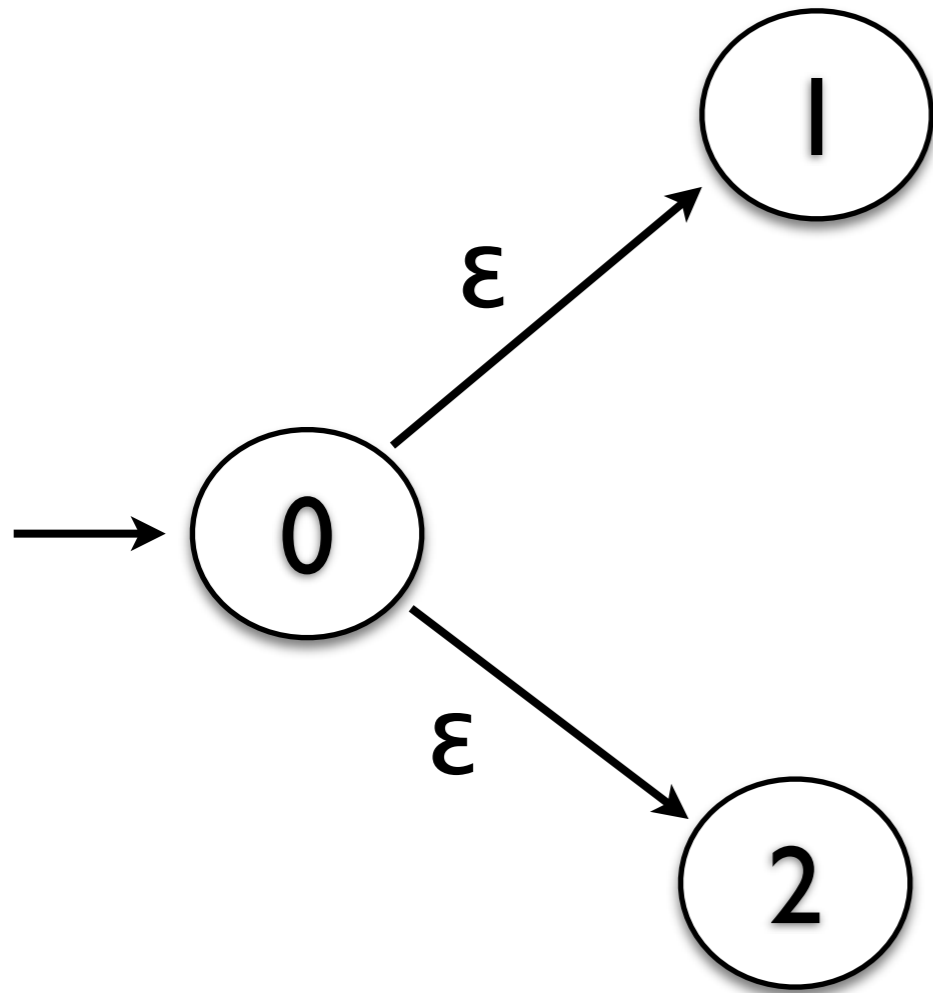
Q1.1: drawing the FSA



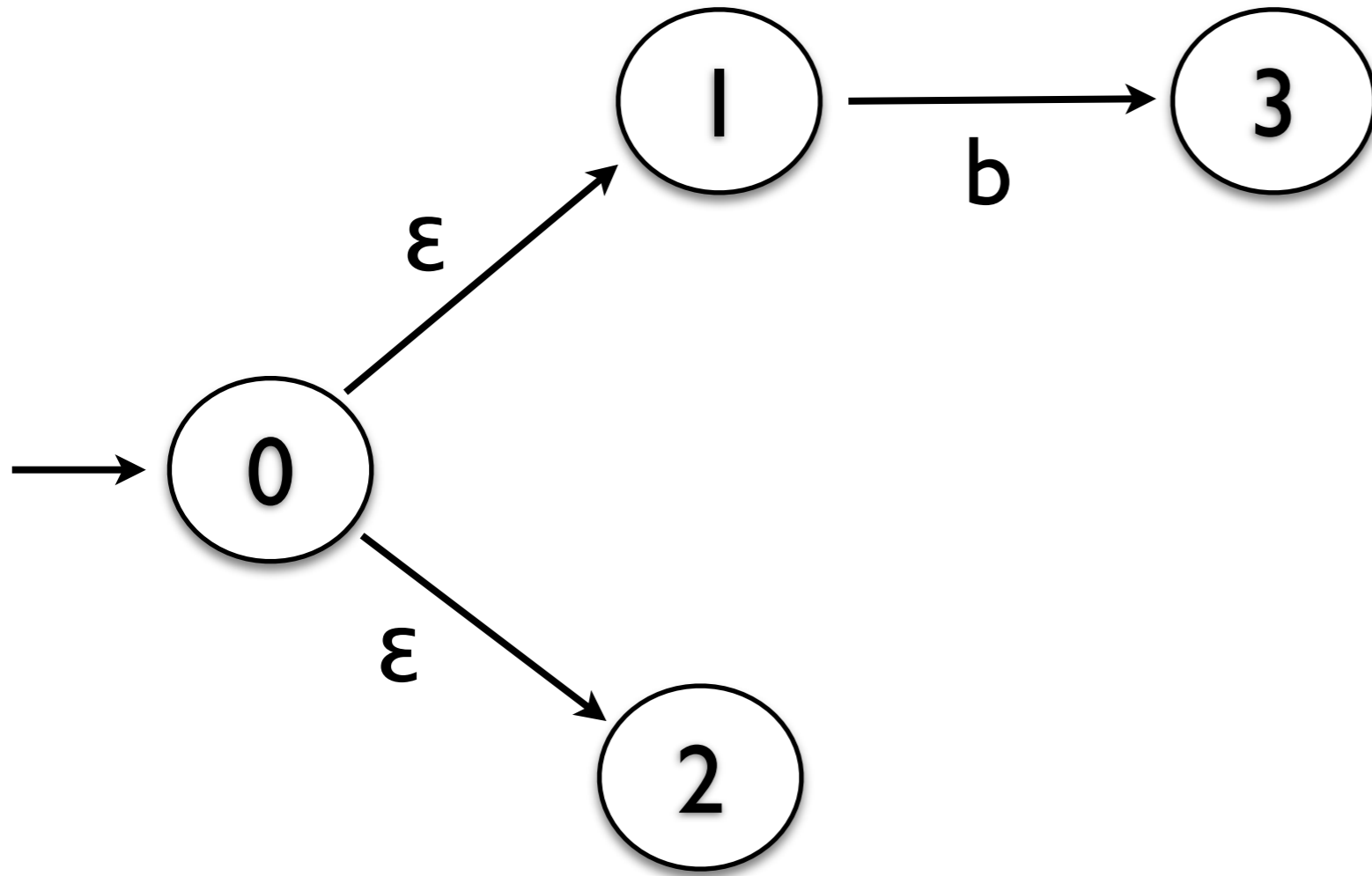
Q1.1: drawing the FSA



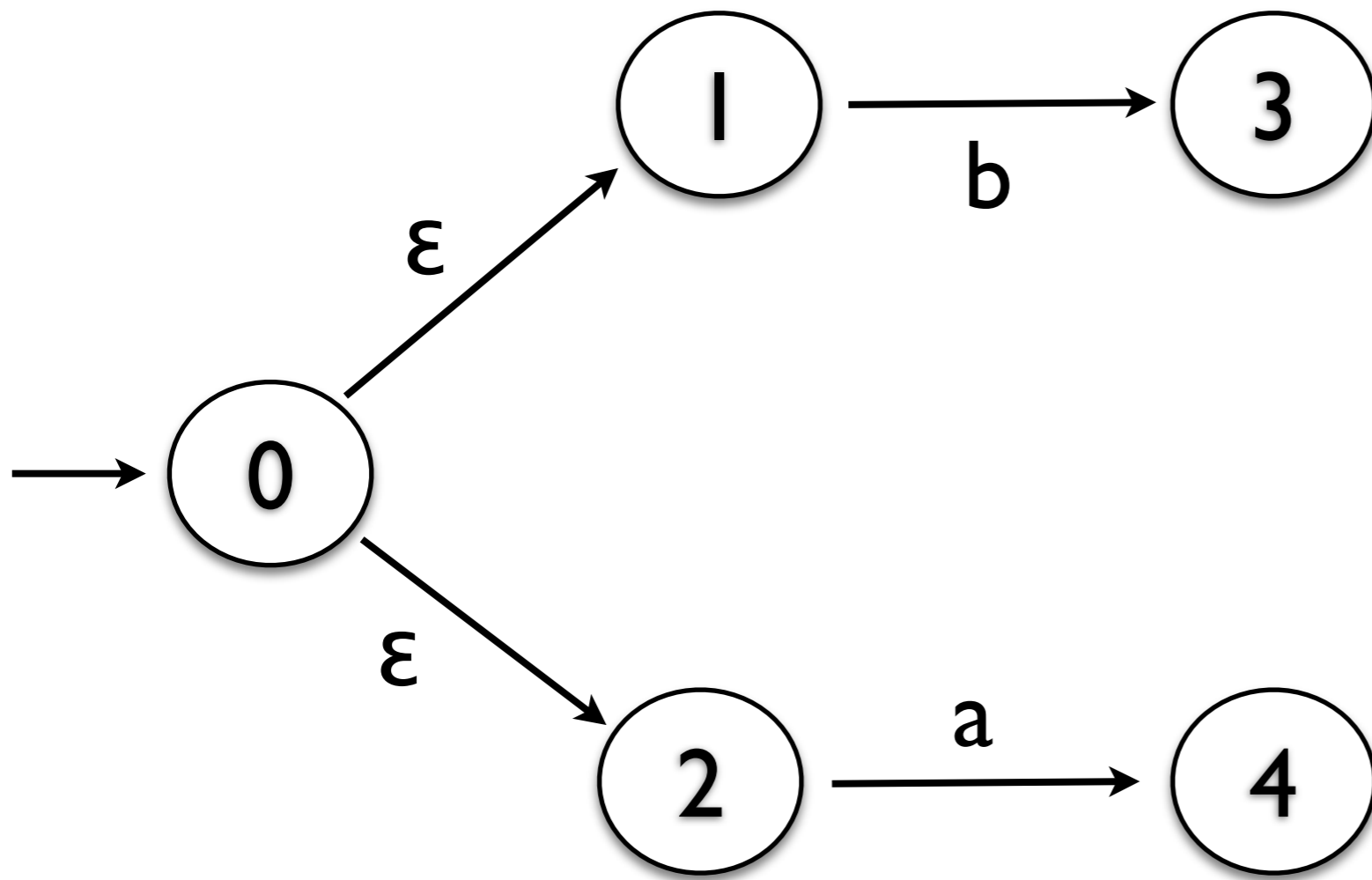
Q1.1: drawing the FSA



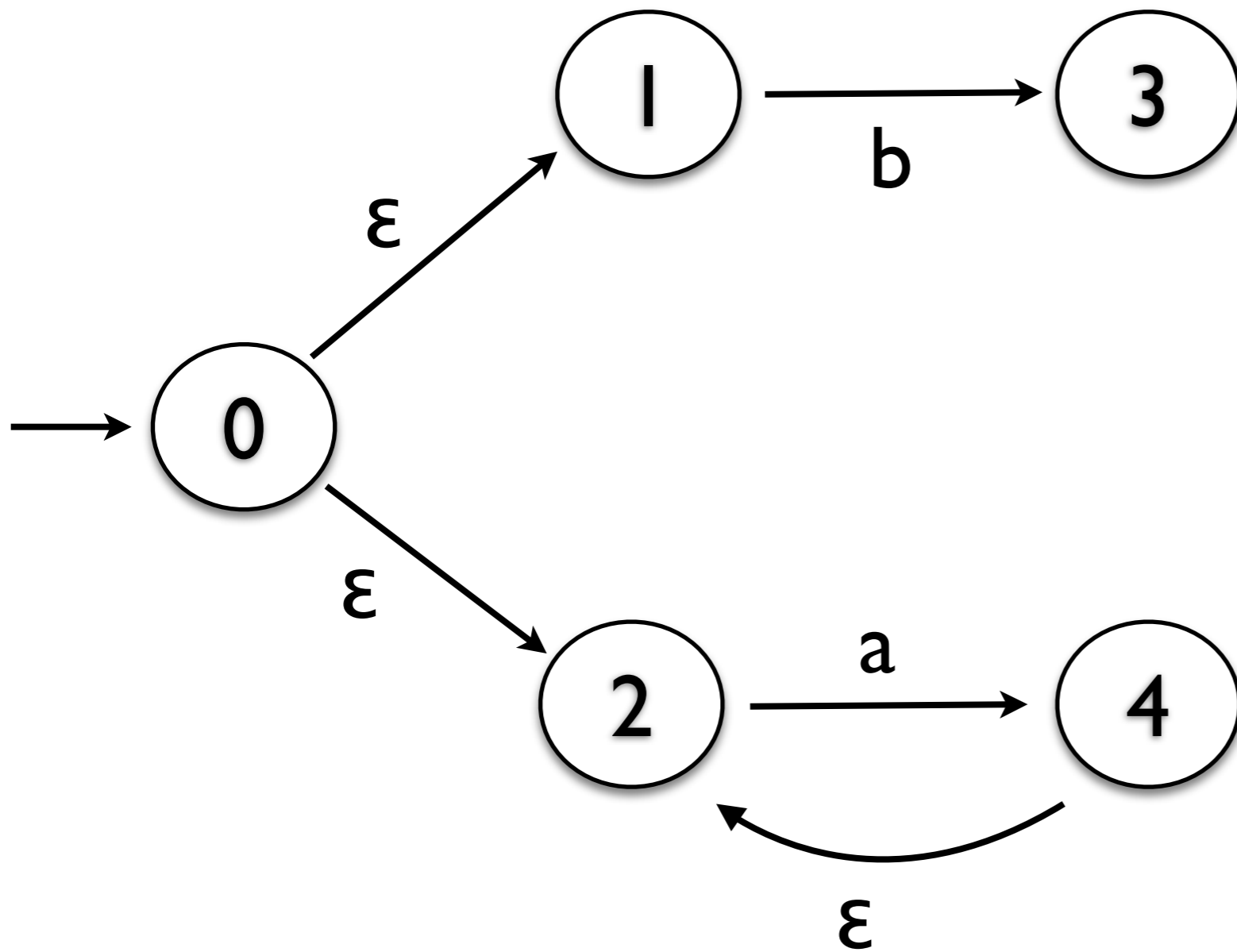
Q1.1: drawing the FSA



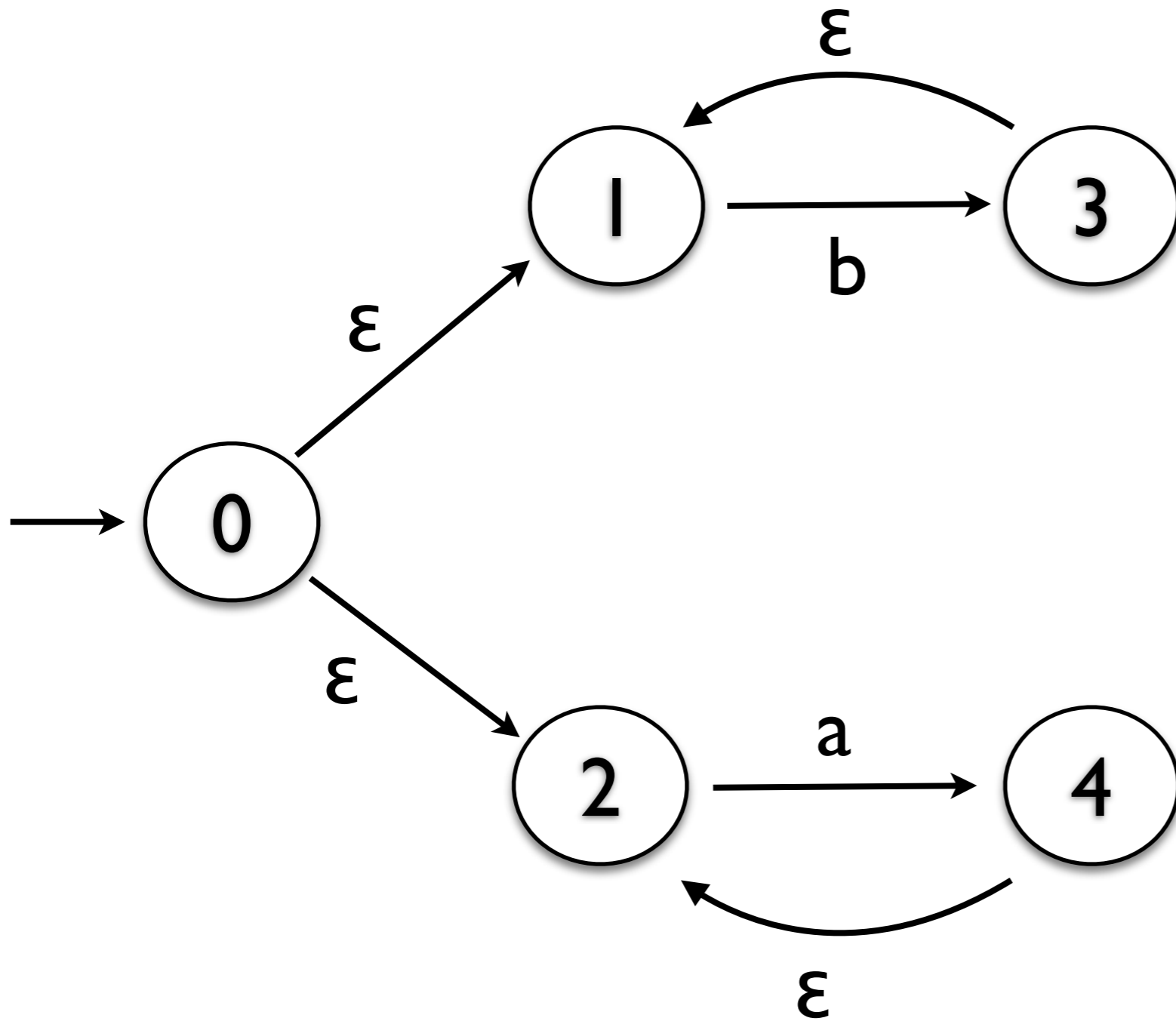
Q1.1: drawing the FSA



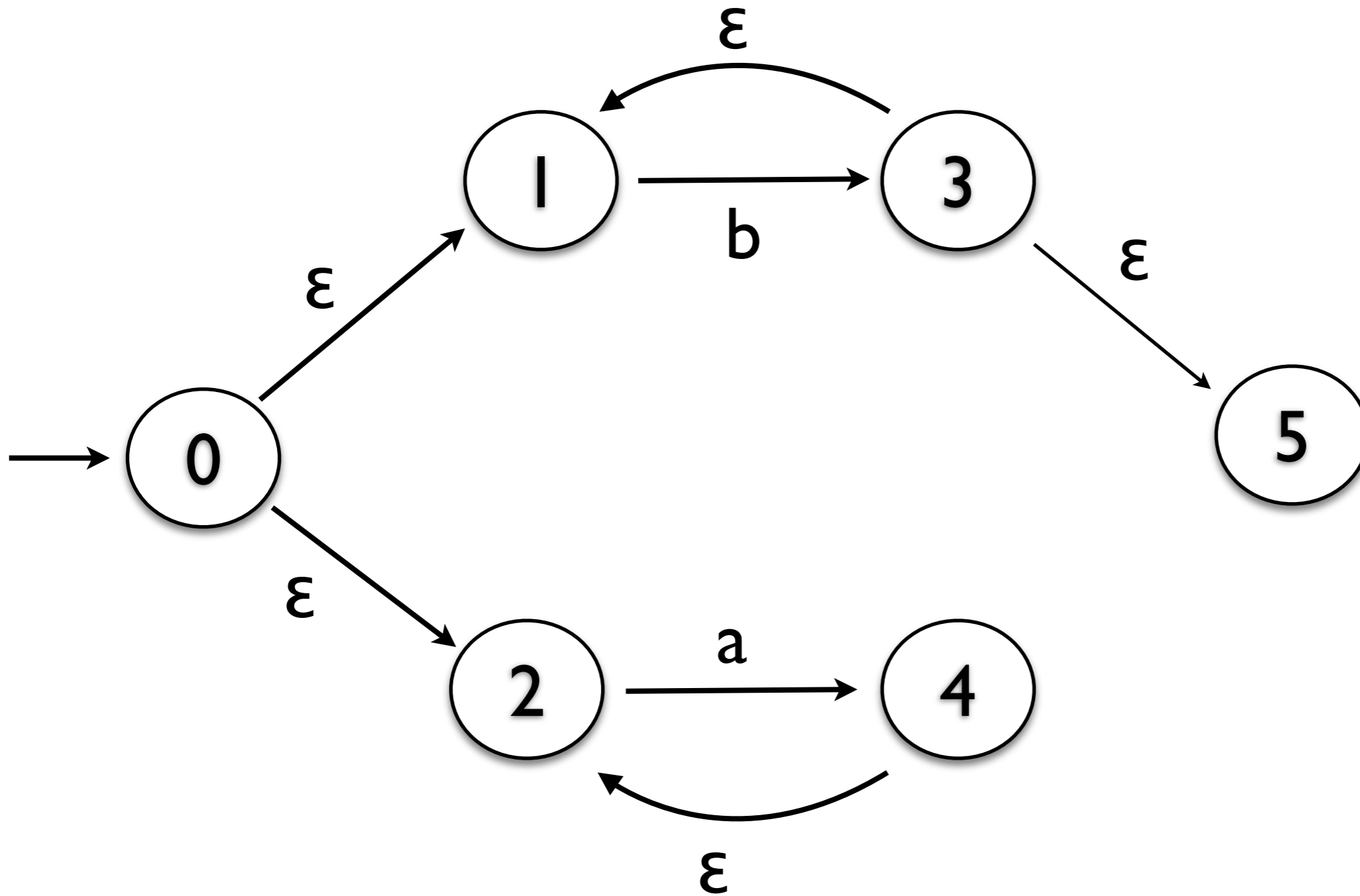
Q1.1: drawing the FSA



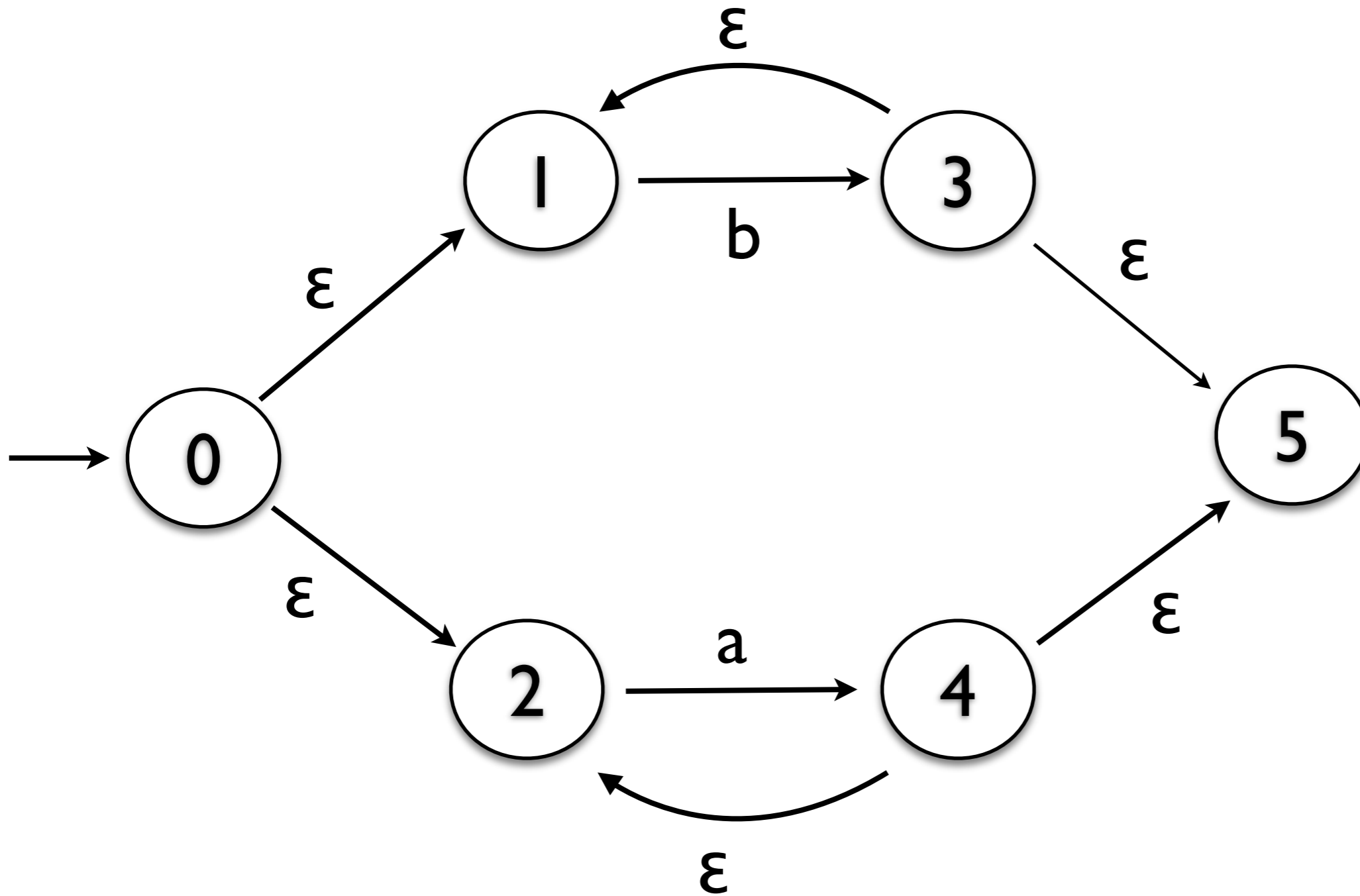
Q1.1: drawing the FSA



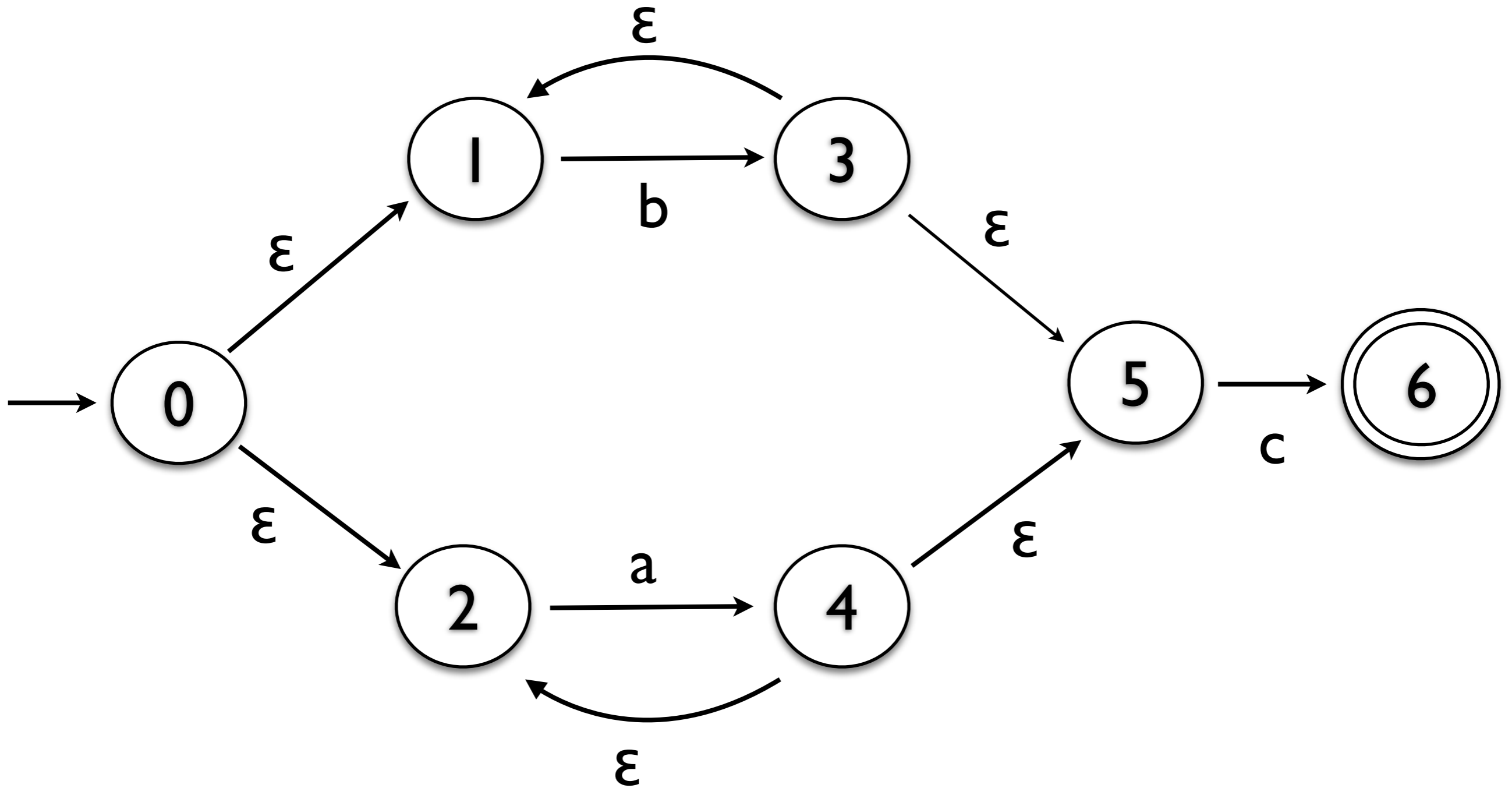
Q1.1: drawing the FSA



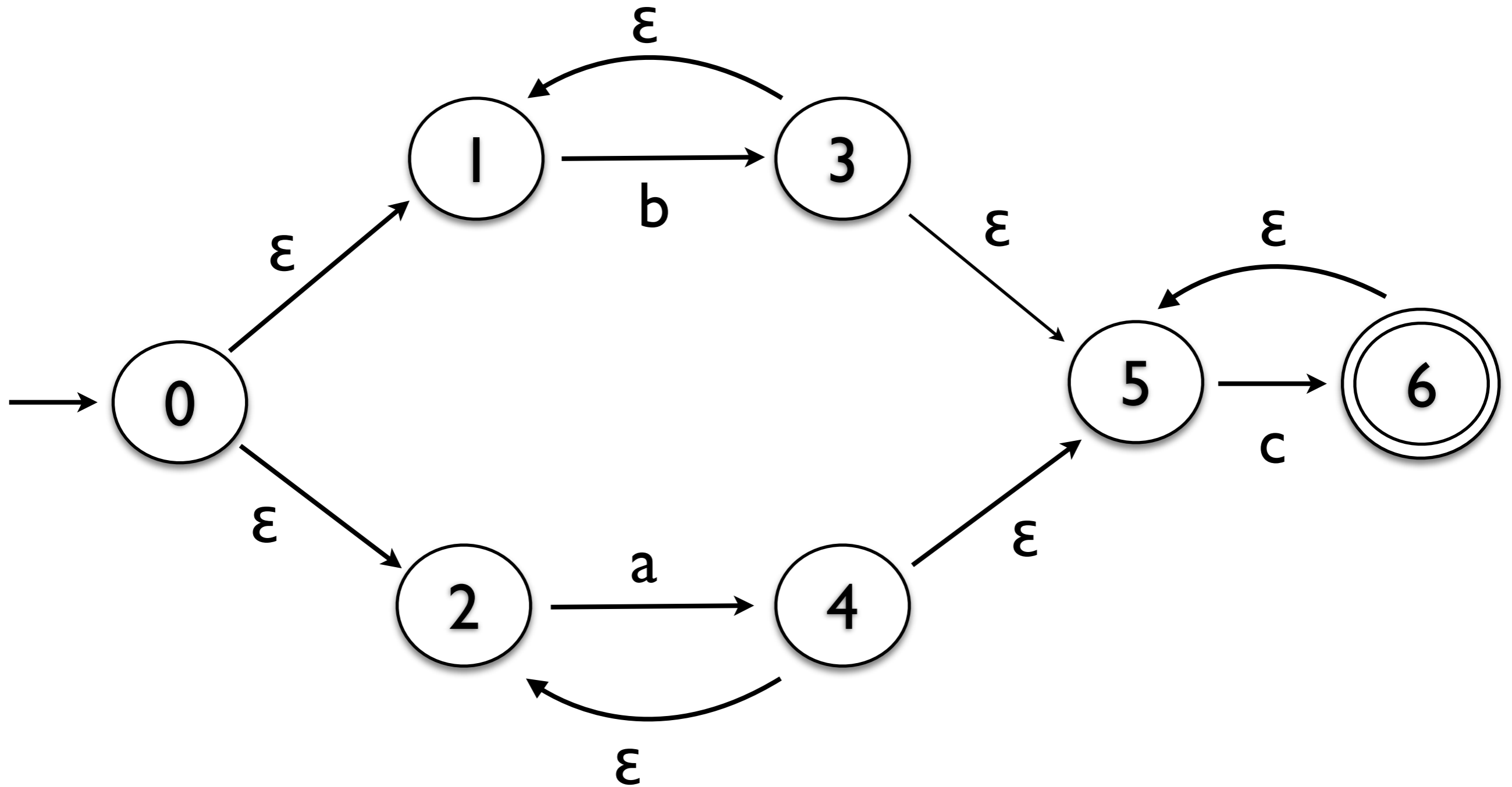
Q1.1: drawing the FSA

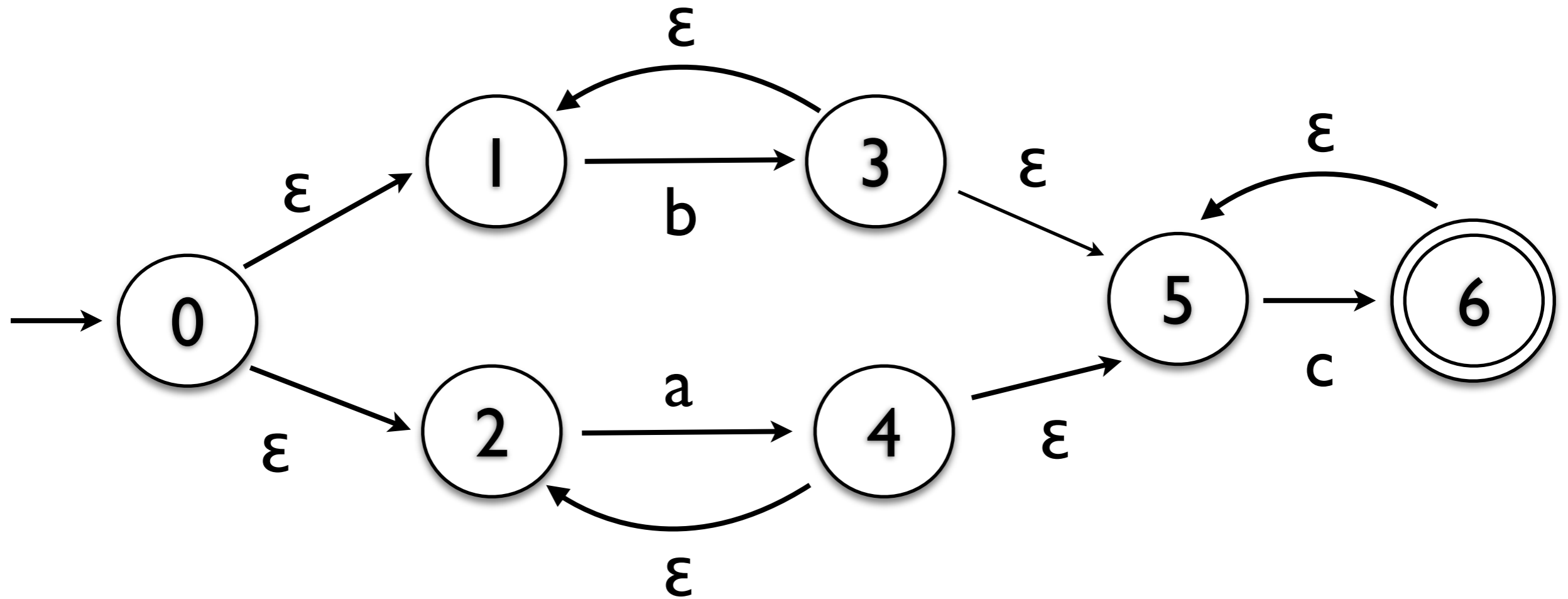


Q1.1: drawing the FSA



Q1.1: drawing the FSA



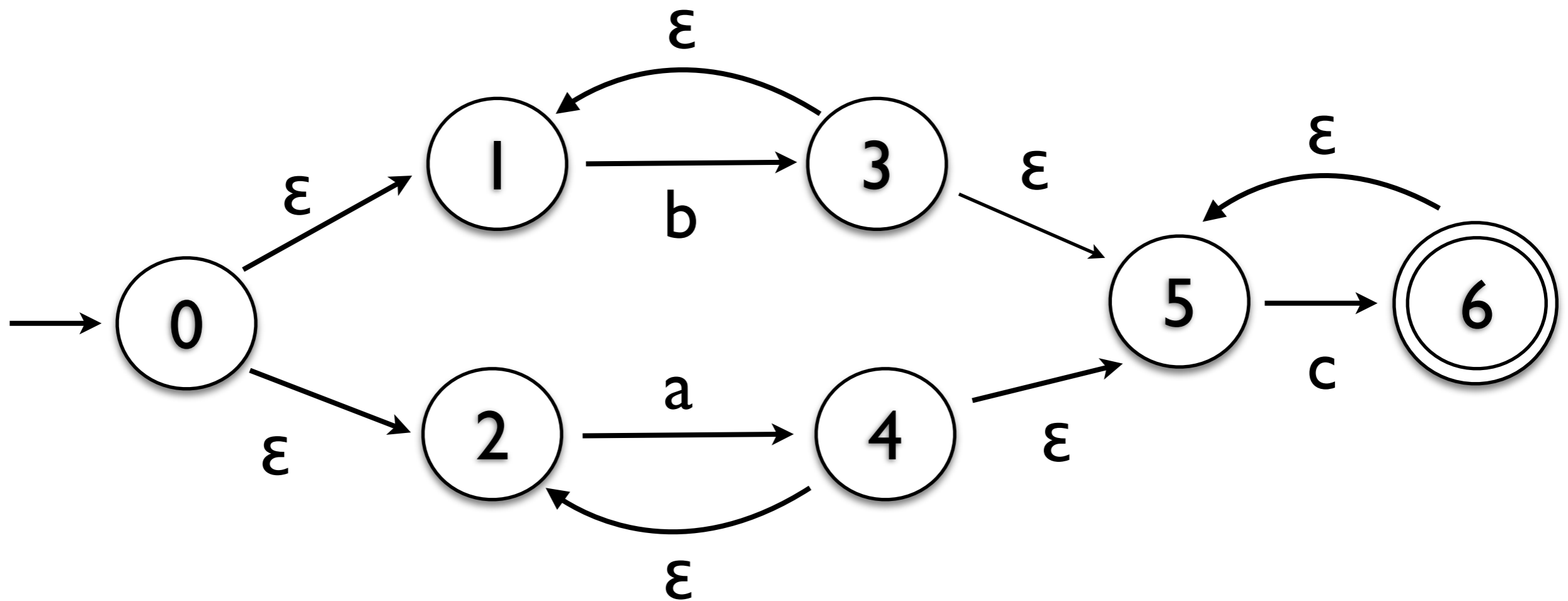


Which language does this automaton accept?

- The subgraph (0-2-4) accepts a^+
- The subgraph (0-1-3) accepts b^+
- (0-1-2-3-4-5) is a union of the two \Rightarrow accepts $a^+|b^+$
- The full FSA accepts $(a^+|b^+)c^+$



- We can start by computing the ϵ -closures, since we'll need them for the subset construction
 - **Reminder:** ϵ -closure of a state $q = q +$ all states that can be reached from q through a sequence of ϵ -transitions
 - ϵ -closure(0) = {0,1,2}, ϵ -closure(1) = {1}, ϵ -closure(2) = {2}, ϵ -closure(3) = {1,3,5}, ϵ -closure(4) = {2,4,5}, ϵ -closure(5) = {5}, ϵ -closure(6) = {5,6}

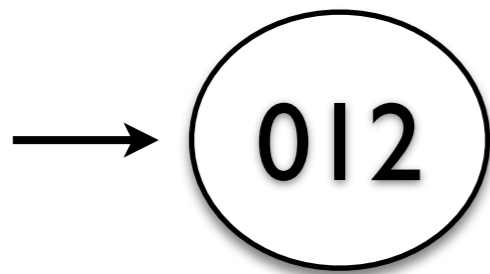




- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012,b) = \{1,3\}$

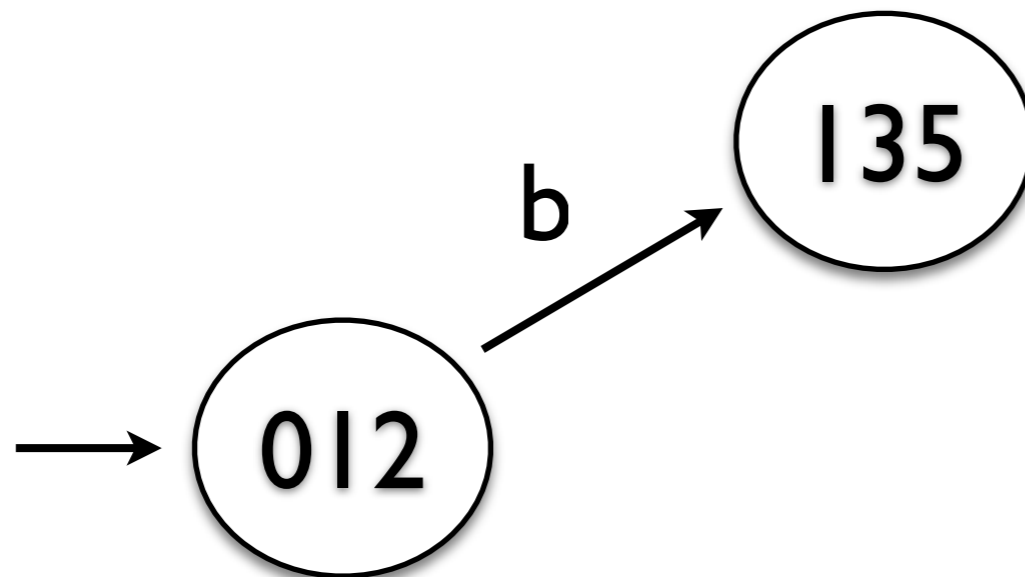


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



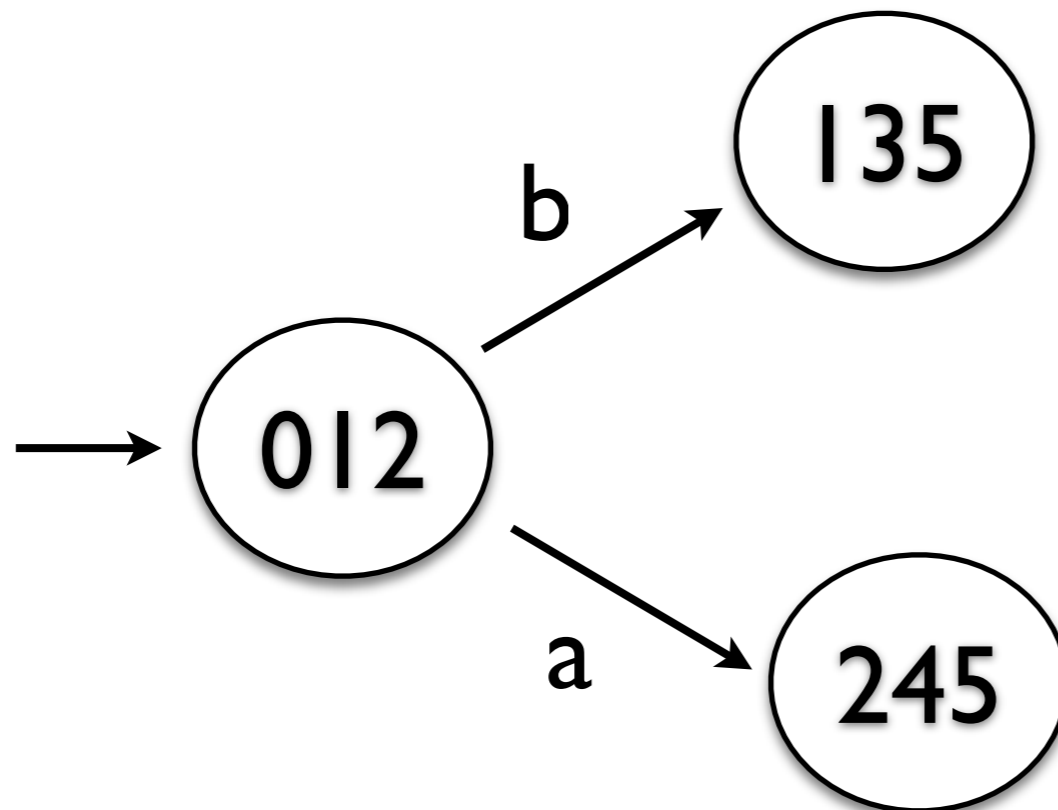


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



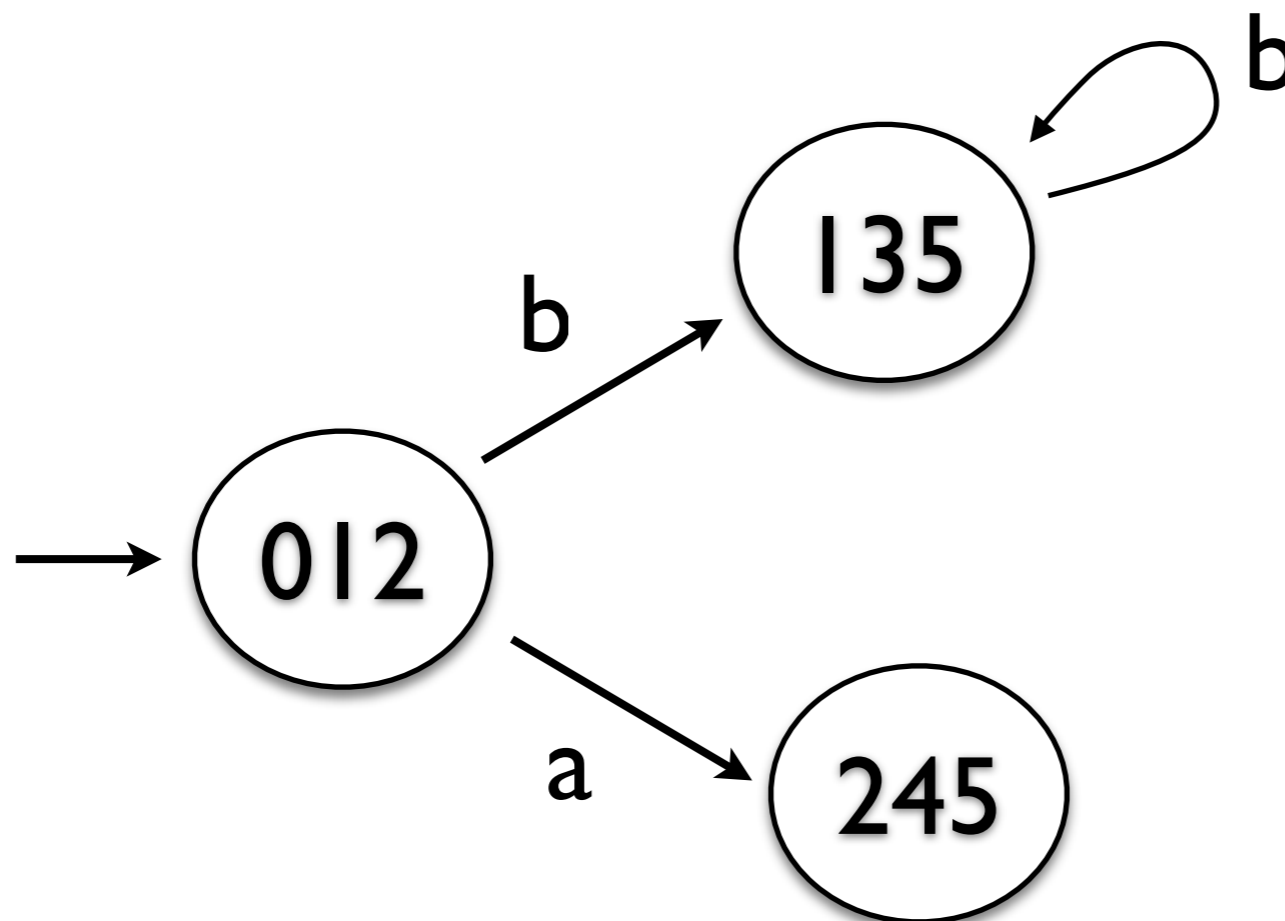


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



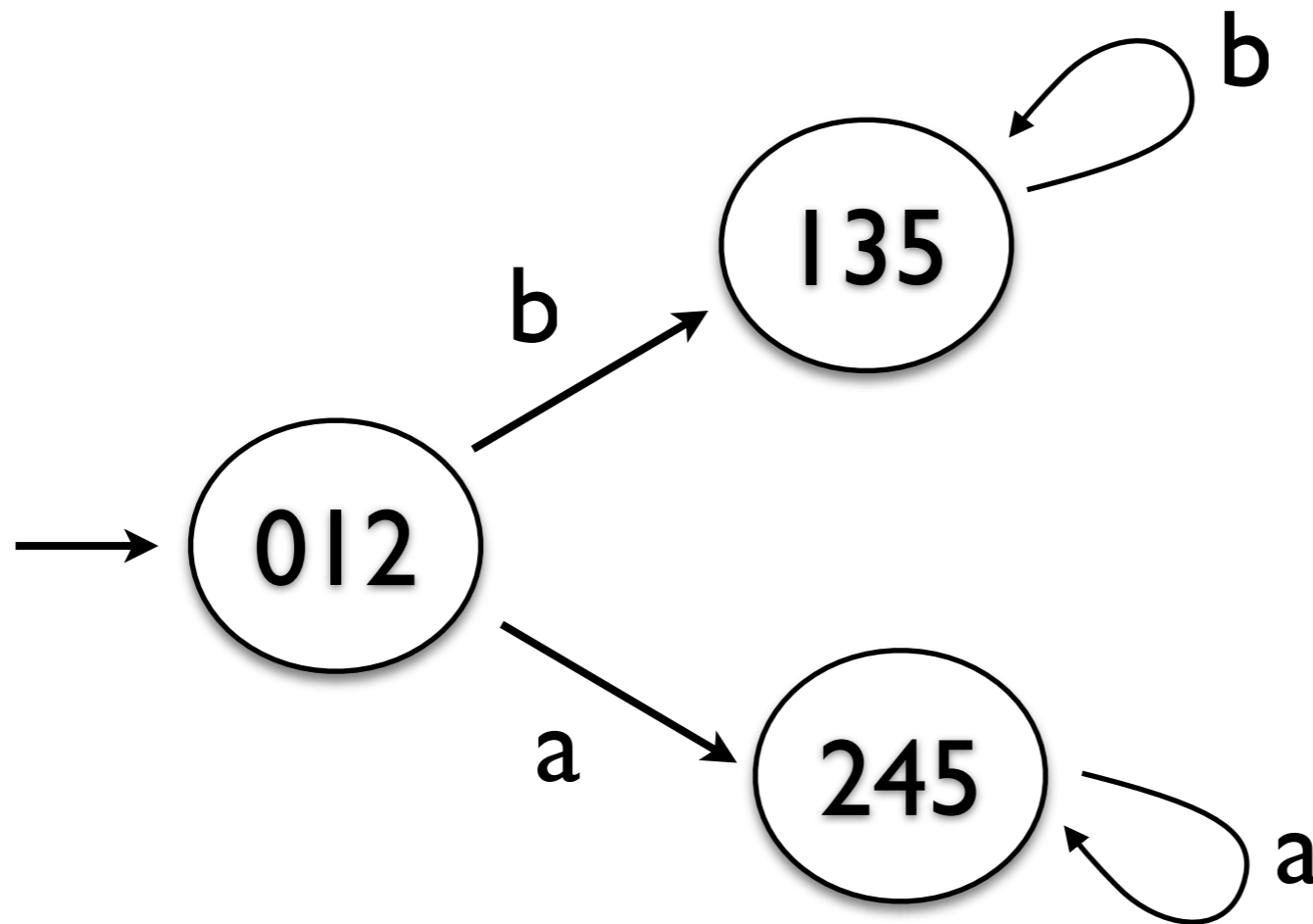


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



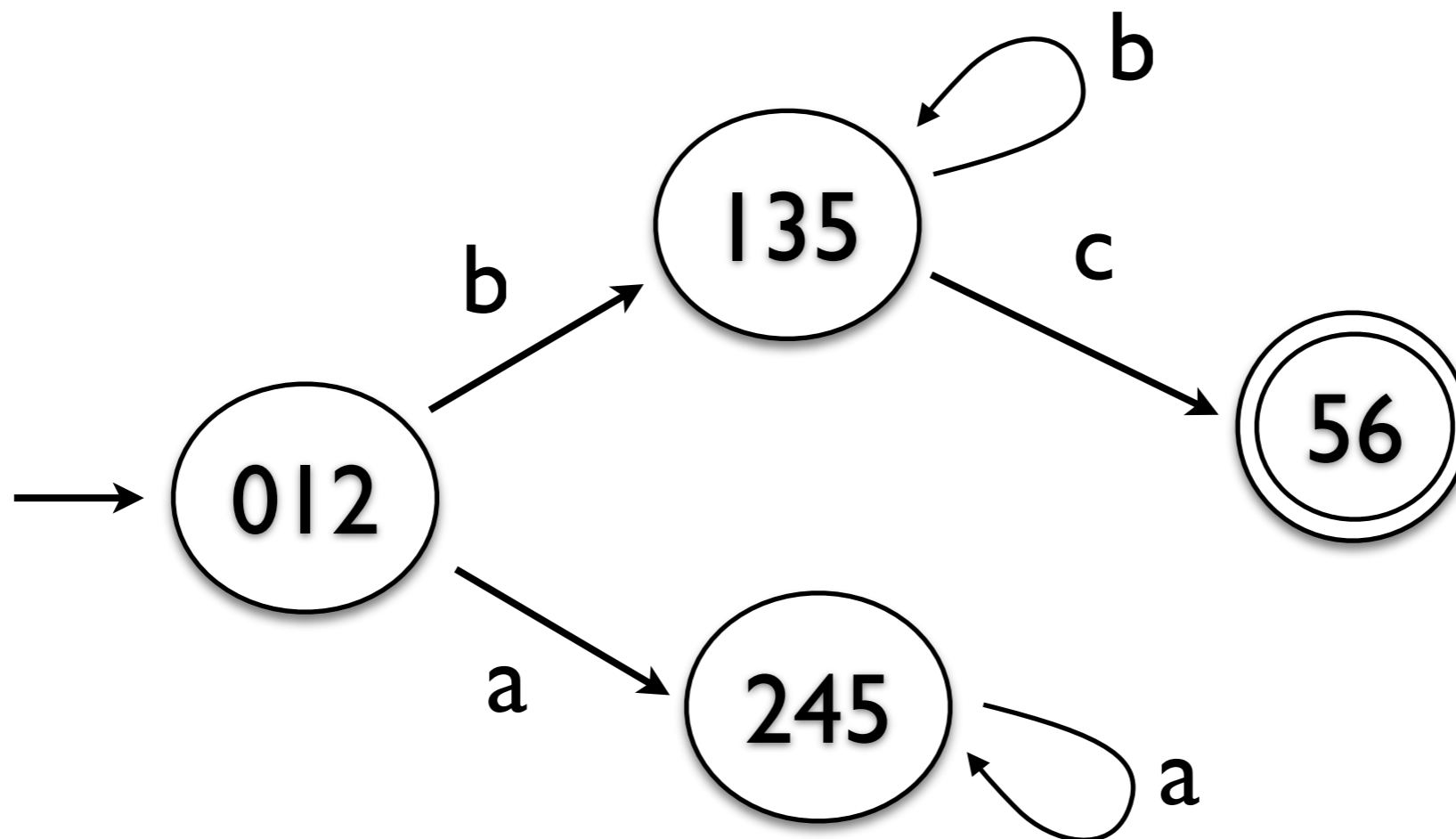


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



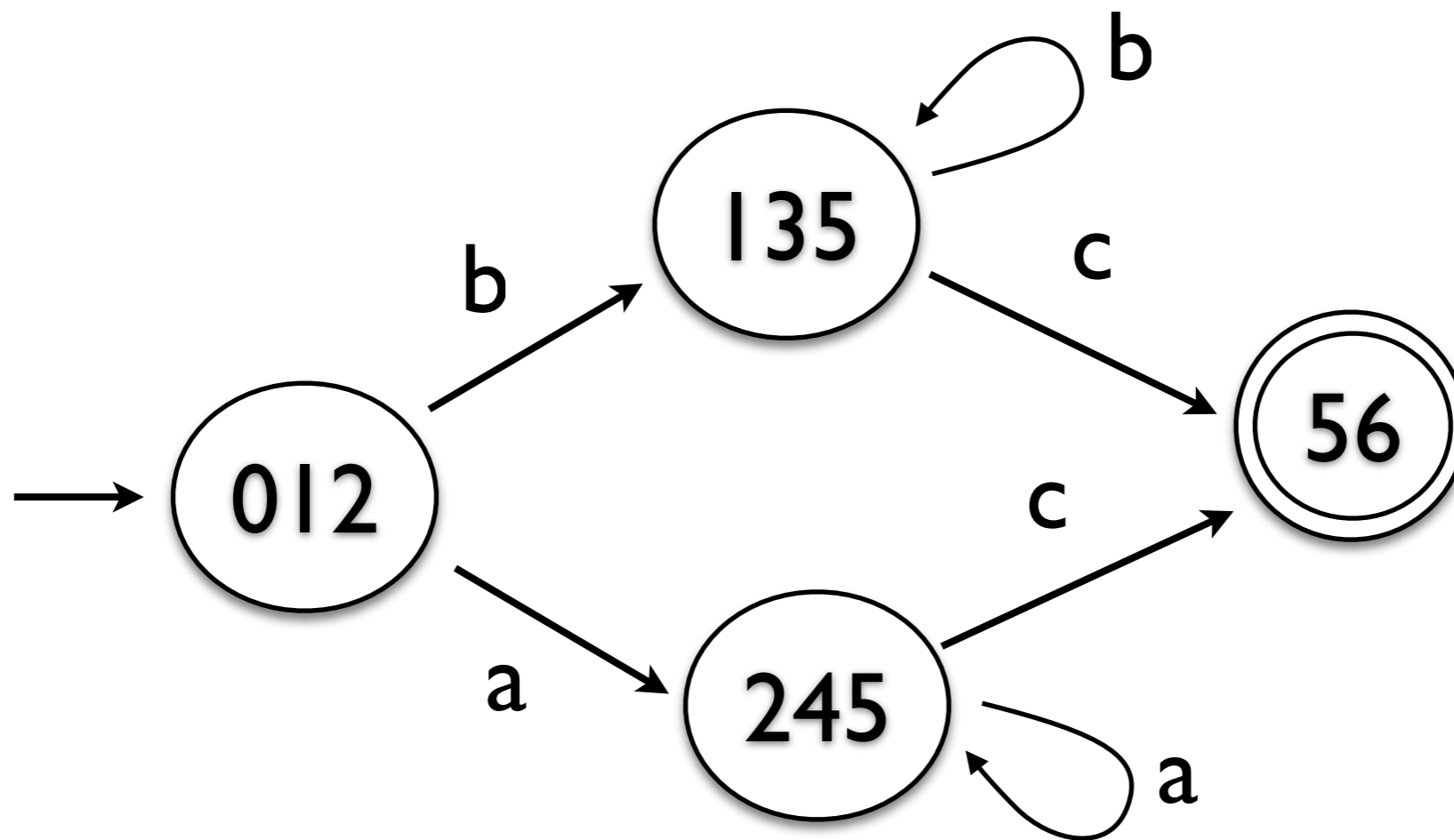


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



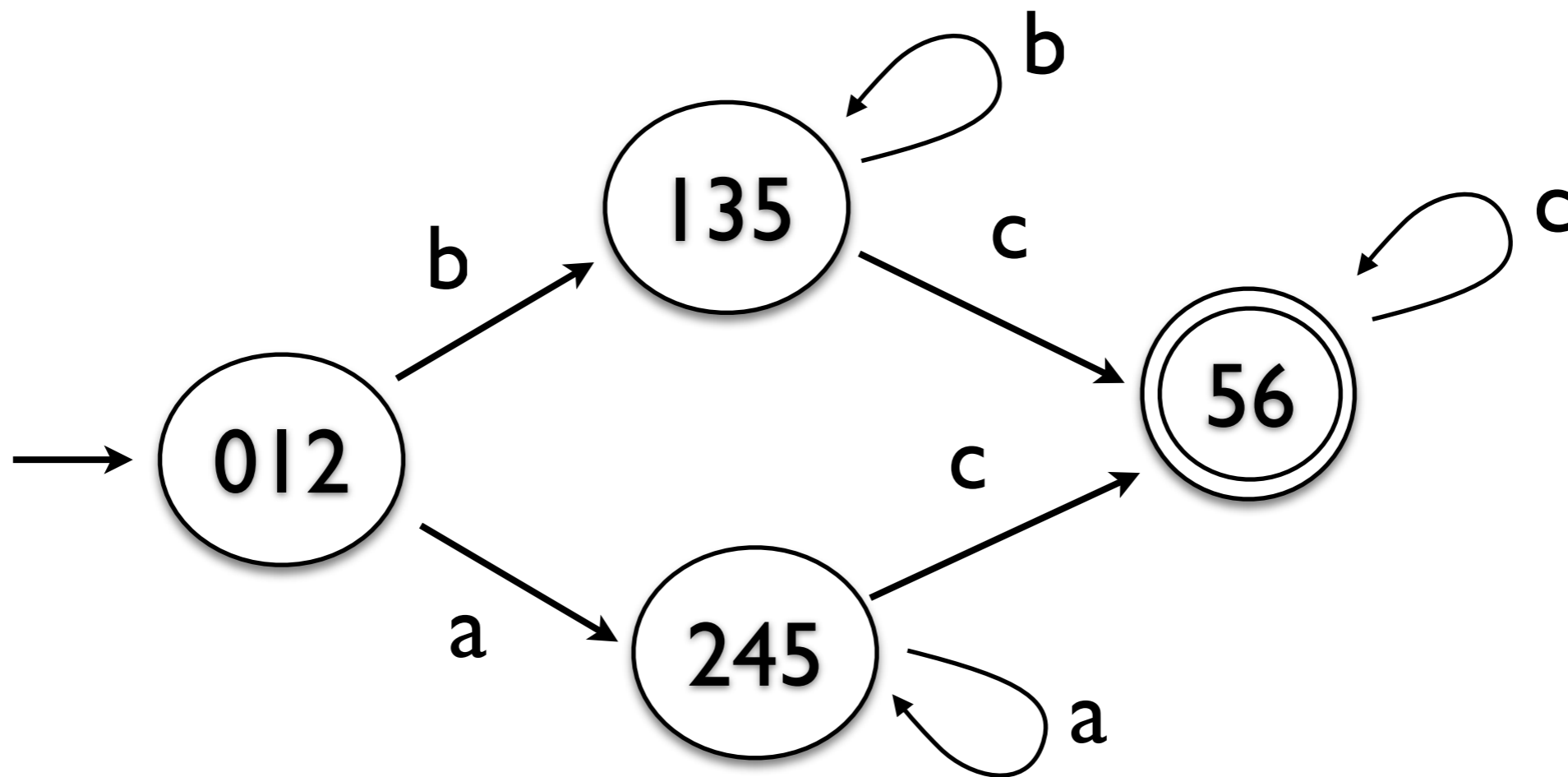


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012, b) = \{1, 3\}$



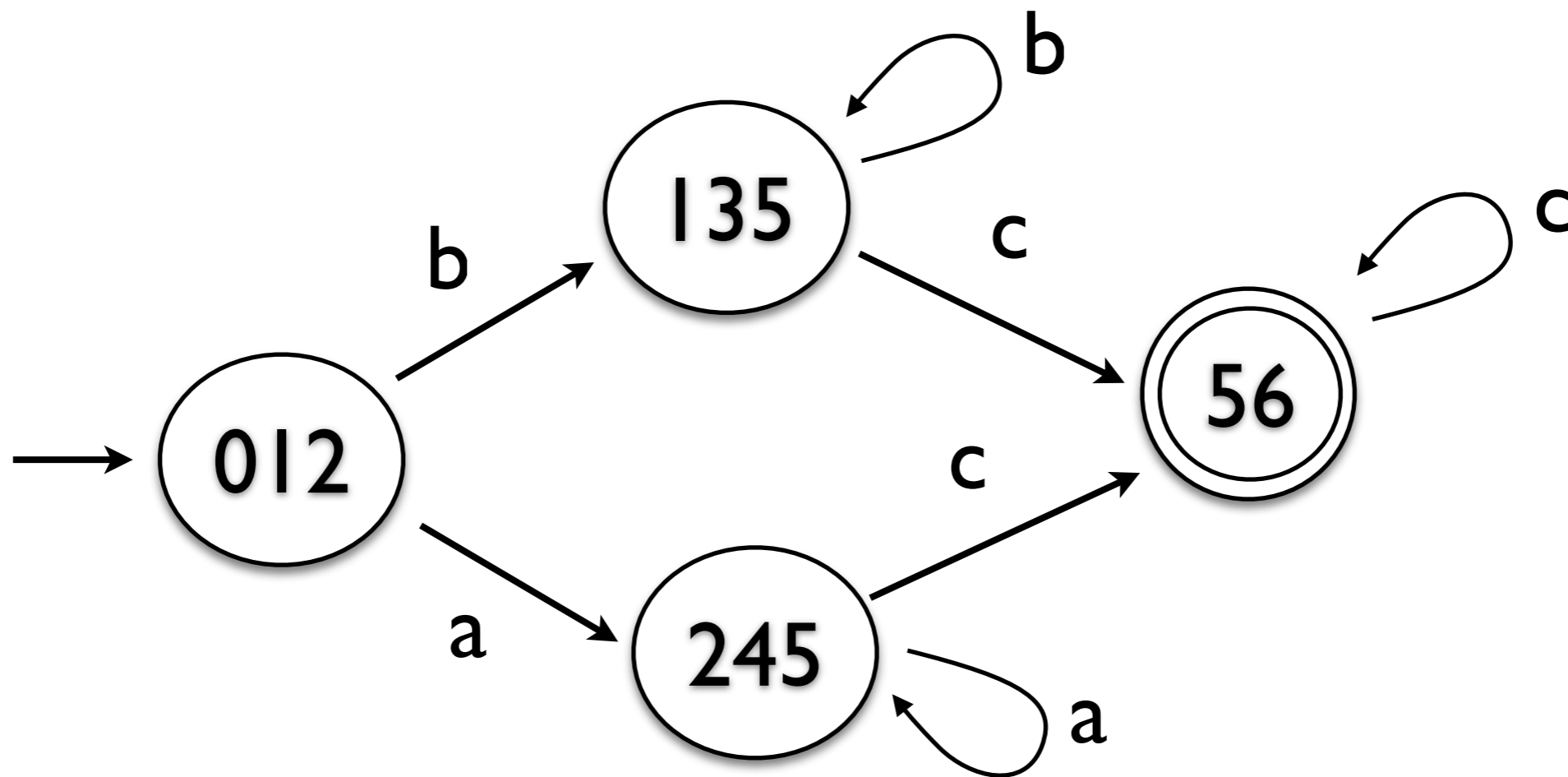


- We then perform the subset construction, starting with the node “012” (ϵ -closure of 0)
- For each constructed node, we look for all the possible states which can be reached from it with a particular character
 - For instance, $\delta'(012,b) = \{1,3\}$





- As a final step, we can verify that the language recognised by the FSA is the same as the non-deterministic one
 - In this case: OK! The language recognised is also $(b^+|a^+)c^+$
 - So we're done :-)





- Provide a formal definition of a Context-Free Grammar
- Explain why the use of a chart can increase parsing efficiency
- What are the advantages and disadvantages of CYK and Earley parsing? Compare these two algorithms, notably in terms of worst-case runtime complexity
- What is the motivation behind the use of unification-based grammars?



- A Context-free grammar is formally defined as a tuple $\langle N, \Sigma, S, P \rangle$ where:
 - N is a set of non-terminal symbols
 - Σ is a set of terminal symbols
 - S is the start symbol $\in N$
 - P is a set of production rules of the form $A \rightarrow \alpha$, where A is a non-terminal and $\alpha \in (\Sigma \cup N)^*$

- A Context-free grammar is formally defined as a tuple $\langle N, \Sigma, S, P \rangle$ where:
 - N is a set of non-terminal symbols
 - Σ is a set of terminal symbols
 - S is the start symbol $\in N$
 - P is a set of production rules of the form $A \rightarrow \alpha$, where A is a non-terminal and $\alpha \in (\Sigma \cup N)^*$



don't forget to mention the form of the production rules! Without such constraints, the grammar could be any grammar of the Chomsky hierarchy.



- *Q2.2: Why the use of a chart can increase parsing efficiency?*
- First question to ask yourself: what is a parse chart?
 - One cell in the chart stores if the substring of the input that is given by the span (start,end) is derivable by stored nonterminals, but not the derivations itself, which can be many.
 - Thus, the subderivations are stored in a compact way and can be used to compute bigger and bigger derivations
- These are two sources of efficiency improvement:
 - no need to recompute subderivations because they are stored and accessible in an efficient way
 - compact storage of subderivations

- *Q2.2: Why the use of a chart can increase parsing efficiency?*
- First question to ask yourself: what is a parse chart?
 - One cell in the chart stores if the substring of the input that is given by the span (start,end) is derivable by stored nonterminals, but not the derivations itself, which can be many.
 - Thus, the subderivations are stored in a compact way and can be used to compute bigger and bigger derivations
- These are two sources of efficiency improvement:
 - no need to recompute subderivations because they are stored and accessible in an efficient way
 - compact storage of subderivations

⇒ Idea of **dynamic programming**: solve a large problem by combining the solutions to various *smaller* subproblems



- Q2.3: Pros and cons of CYK and Earley parsing

CYK	Earley
<ul style="list-style-type: none">• Bottom-up parser• Explore all possible derivations• Good for robust parsing (extraction of chunks)• Polynomial complexity: $O(G ^2n^3)$• (also easier to use when dealing with lexicalised grammar)	<ul style="list-style-type: none">• Top-down parser• Not all sub-derivations are computed• Difficult to apply for robust parsing• Complexity: $O(n^3)$ in general, $O(n^2)$ for unambiguous grammars• Usually better average runtime



- *Q2.4: Explain the motivation behind the use of unification-based grammars*
- Generalizations such as the agreement between subject and verb can not easily be expressed in CFGs
 - The number of nonterminals would have to grow massively, as well as the lexical ambiguity.
 - Furthermore, such grammars would not be manageable anymore and very hard to test.
- View grammatical categories as *objects* which can have complex set of *properties* attached to them - instead of simple atomic primitives
- More fine-grained way of *representing* and *placing constraints* (in the rules) on the grammatical categories



- Q3: What is meant by the open-world assumption in description logic/OWL? Provide an example.
- **Open-world assumption:** what can NOT proven to be true is NOT believed to be false
- For instance, not knowing whether a student passed his exam doesn't mean that he didn't pass it :-)
- Example with an ontology:
 - {Woman(alice), hasChild(alice, doris), hasChild(alice, boris)}
 - If we ask the question: {alice} $\sqsubseteq \leq 2$ hasChild, we can only answer that we don't know... because she might have more than 2 children!
 - Whereas if we ask the question {alice} $\sqsubseteq \geq 2$ hasChild, the answer is then yes (if we assume that doris and boris are distinct individuals)

- Write down the general solution of the maximum entropy model. Explain why evaluating a Maximum Entropy Model is computationally expensive.
- Assume you want to determine the probability of getting a grade ≤ 2.0 for the Computational Linguistics course. You have the following information at your disposal:
 - For each student, you know whether he/she submitted all his/her assignments
 - You know that 60 % of all students get a grade ≤ 2.0
 - And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a Maximum Entropy Model. Write down the set of linear constraints that such model would include. (note: you don't have to compute the final distribution, just provide the constraints)



- *Q4.1: Write down the general solution of the MaxEnt model. Explain why evaluating such model is computationally expensive.*
- General solution with a log-linear model (see slide 31):

$$p_{\lambda}(y|x) = \frac{1}{Z_{\lambda}(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

- The evaluation of a MaxEnt model is computational expensive because of the normalisation
 - i.e. computing the value of the parameter $Z_{\lambda}(x)$:

$$Z_{\lambda}(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:

- For each student, you know whether he/she submitted all his/her assignments
- You know that 60 % of all students get a grade ≤ 2.0
- And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- Set y to be the outcome of getting a grade ≤ 2 , with the domain $y \in [T, F]$
- And set x to be the evidence variable about submitting all the assignments, with domain $x \in [T, F]$
- We are interested in the probability $P(y|x)$
- Intuitively, we would like to express the following constraints:
 - $P(y=T, x=T) + P(y=T, x=F) = 0.6$
 - $P(y=T \mid x=T) = 0.9$
 - $P(y=T, x=F) + P(y=T, x=T) + P(y=F, x=T) + P(y=F, x=F) = 1.0$

- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:
 - For each student, you know whether he/she submitted all his/her assignments
 - You know that 60 % of all students get a grade ≤ 2.0
 - And you know that 90% of the students who submitted all their assignments get a grade ≤ 2


You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- *How do we specify these constraints in a MaxEnt model?*
- Ganz einfach:
 - For the constraint $P(y=T, x=T) + P(y=T, x=F) = 0.6$, we create a feature function f_1 defined as

$$f_1(x, y) = \begin{cases} 1 & \text{if } y=\text{true} \\ 0 & \text{otherwise} \end{cases}$$

- For the constraint $P(y=T|x=T) = 0.9$, we create a feature function f_2 defined as

$$f_2(x, y) = \begin{cases} 1 & \text{if } y=\text{true} \text{ and } x=\text{true} \\ 0 & \text{otherwise} \end{cases}$$

- 
- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:

- For each student, you know whether he/she submitted all his/her assignments
- You know that 60 % of all students get a grade ≤ 2.0
- And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- *How do we specify these constraints in a MaxEnt model?*
- Remember that a constraint is used to match the model to the known statistics, i.e.

$$p(f_i) = \tilde{p}(f_i)$$

- Or in other words, with linear constraints:

$$\sum_{x,y} \tilde{p}(x)p(y|x)f_i(x,y) = \sum_{x,y} \tilde{p}(x,y)f_i(x,y)$$

- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:
 - For each student, you know whether he/she submitted all his/her assignments
 - You know that 60 % of all students get a grade ≤ 2.0
 - And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- *How do we specify these constraints in a MaxEnt model?*
- For the constraint $P(y=T, x=F) + P(y=T, x=T) = 0.6$, we thus have:

$$\sum_{x,y} \tilde{p}(x)p(y|x)f_1(x,y) = \sum_{x,y} \tilde{p}(x,y)f_1(x,y)$$

$$\sum_{x,y} \tilde{p}(x)p(y|x)f_1(x,y) = 0.6$$

$$\sum_x \tilde{p}(x)p(\text{true}|x) = 0.6$$

- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:
 - For each student, you know whether he/she submitted all his/her assignments
 - You know that 60 % of all students get a grade ≤ 2.0
 - And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- *How do we specify these constraints in a MaxEnt model?*
- For the constraint $P(y=T|x=T) = 0.6$, we thus have:

$$\sum_{x,y} \tilde{p}(x)p(y|x)f_2(x,y) = \sum_{x,y} \tilde{p}(x,y)f_2(x,y)$$

$$\sum_{x,y} \tilde{p}(x)p(y|x)f_2(x,y) = \tilde{p}(T,T) = \tilde{p}(T|T) \tilde{p}(T) = 0.9 \tilde{p}(T)$$

$$\tilde{p}(T)p(T|T) = 0.9 \tilde{p}(T) \Leftrightarrow p(T|T) = 0.9$$



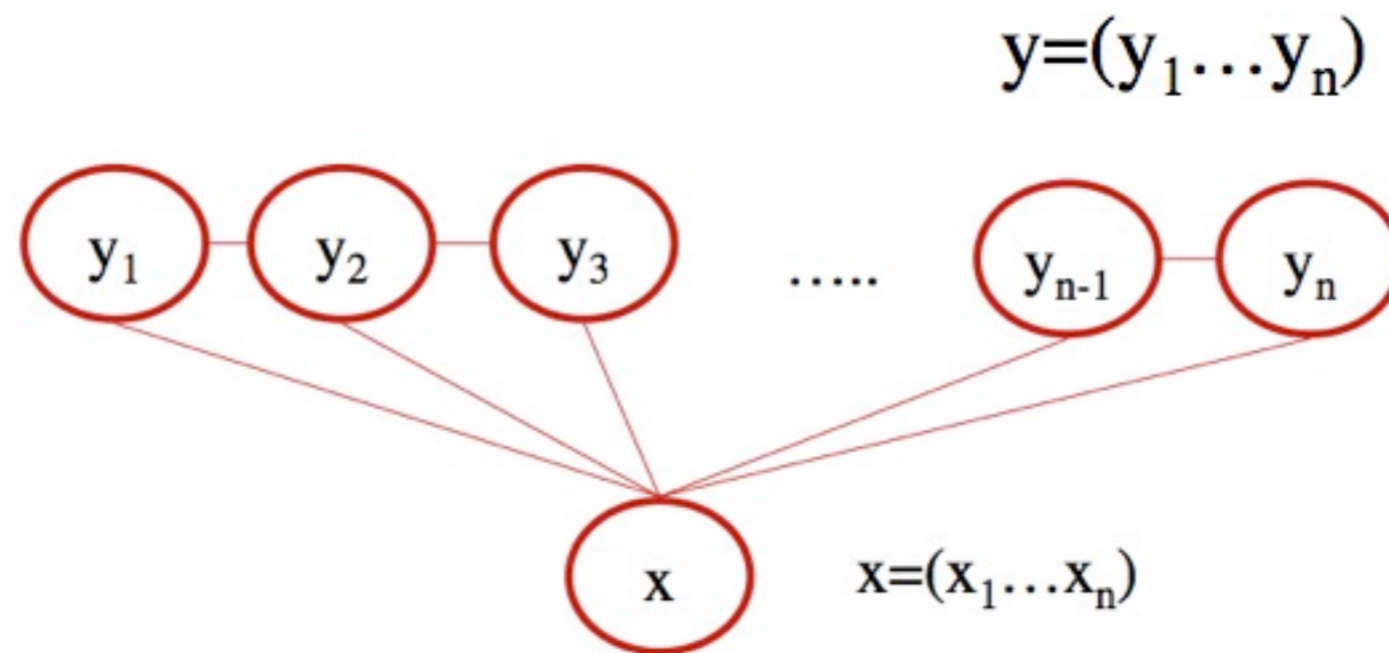
- You want to determine the probability of getting a grade ≤ 2.0 for the CL course. You have the following information at your disposal:
 - For each student, you know whether he/she submitted all his/her assignments
 - You know that 60 % of all students get a grade ≤ 2.0
 - And you know that 90% of the students who submitted all their assignments get a grade ≤ 2

You would like to determine the probability $P(\text{getting a grade } \leq 2 \mid \text{submitted all his/her assignments})$, using a MaxEnt Model. Write down the set of linear constraints that such model would include.

- *How do we specify these constraints in a MaxEnt model?*
- ... and we can do the same for the last constraint (for the normalisation)



- *Q: Provide a graphical representation of a Conditional Random Field. Which algorithm is used to find the most likely hidden sequence, and what is its runtime complexity?*
- Graphical illustration of a CRF:



Each random variable y_i is conditioned on the complete input sequence x_1, \dots, x_n



- *Q: Provide a graphical representation of a Conditional Random Field. Which algorithm is used to find the most likely hidden sequence, and what is its runtime complexity?*

The algorithm used to find the most likely hidden sequence is **Viterbi** decoding

The matrix M replaces the product of transition and emission probability

Runtime complexity:

- linear in length of sequence
- quadratic in the number of labels



- Explain why the Boyer-Moore algorithm may yield sublinear computing time.
- Use the Boyer-Moore algorithm to find matches of the pattern $P = \text{ADCDCD}$ in the text $T = \text{ABDADCDCDABABBUDCBACDA}$



- *Q6.1: Explain why the Boyer-Moore algorithm may yield sublinear computing time.*
- Boyer-Moore may be sublinear thanks to the *bad character* and *good suffix* rules
 - For the bad character rule: upon a mismatch in the character comparisons (with mismatched character in $T = x$), shift the pattern to the right-most x in P
 - In other words, not all characters in T need to be compared!

- *Q6.2: Use the Boyer-Moore algorithm to find matches of the pattern $P = ADCDXCD$ in the text $T = ABDADCDXCDABABBUDCBACDA$*

P= 1 2 3 4 5 6 7
 A D C D X C D

- We first do the necessary preprocessing on the pattern P , to compute $L'(i)$ and $I'(i)$
 - in particular, we have $L'(7) = 2$ $L'(6) = 4$
 - $I'(i) = 0$ for every i



T= A B D A D C D X C D A B A B B U D C B A C D A

P= A D C D X C D

X ✓ ✓

→ Use of the good suffix rule



T= A B D A D C D X C D A B A B B U D C B A C D A

P= A D C D X C D

 ✓ ✓ ✓ ✓ ✓ ✓ ✓

→ woohoo!

$l'(2) = 0$, so we completely shift P



T= A B D A D C D X C D A B A B B U D C B A C D A

P= A D C D X C D

X ✓

→ Use of the bad character rule



T= A B D A D C D X C D A B A B B U D C B A C D A
P= A D C D X C D
x

→ and we're done :-)



- Don't hesitate to contact me for any questions
- Needless to say, *don't* assume that material which has not been covered in the Probeklausur won't appear in the real exam