Regular Expressions & Finite State Automata – Part 2
ICS 482: Natural Language Processing

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

Previous Lectures

- 1 Assignment #1
- 1 Pre-start online questionnaire
- 2 Introduction to NLP
- 2 Phases of an NLP system
- 2 NLP Applications
- 3 Chatting with Alice
- 3 Regular Expressions
- 3 Finite State Automata
- 3 Regular languages
- 3 Assignment #2

Objective of Today's Lecture

- Regular Expressions
- Regular languages
- Deterministic Finite State Automata
- Non-deterministic Finite State Automata
- Accept, Reject, Generate terms

Review

- Regular expressions are a textual representation of FSAs
- Recognition is the process of determining if a string/ input is in the language defined by some machine
  – Recognition is straightforward with deterministic machines

Regular Expressions

- Matching strings with regular expressions is a matter of
  - translating the expression into a machine (table) and
  - passing the table to an interpreter

Substitutions in RE

- Substitutions

```
s/colour/color/
```

- Memory (`\1`, `\2`, etc. refer back to matches)

```
s/([0-9]+)/<\1>/
```

- Put angle brackets around all integers
- Practice with Microsoft Word

Eliza-style regular expressions

Eliza is an 'old version' of ALICE.
Step 1: replace first person references with second person references
Step 2: use additional regular expressions to generate replies
Step 3: use scores to rank possible transformations

```
s/I am/You are/
s/I'm/You are/
s/my/your/
```

Eliza-style regular expressions

Eliza is an ‘old version’ of ALICE.

Step 1: replace first person references with second person references

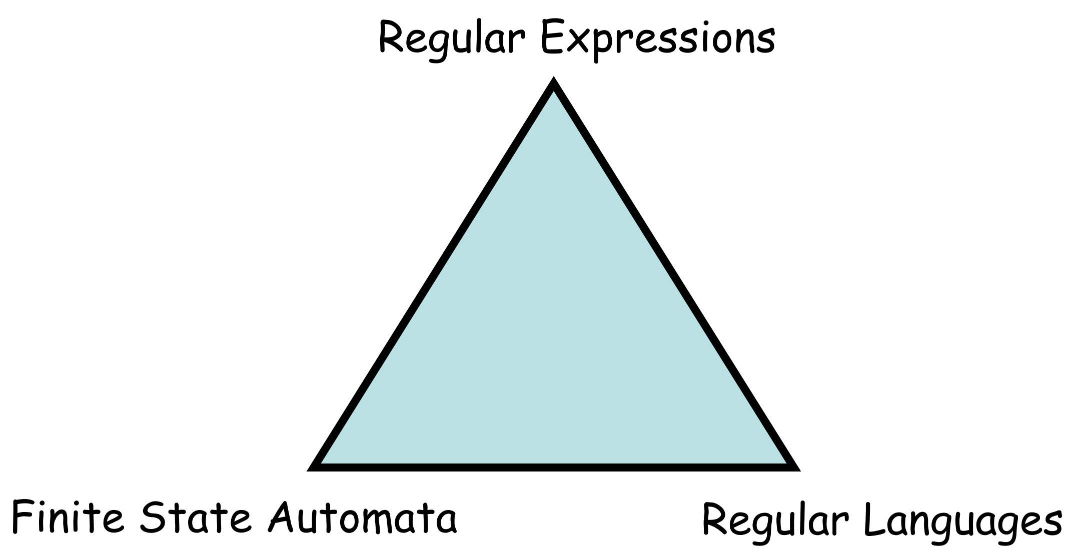
Step 2: use additional regular expressions to generate replies

Step 3: use scores to rank possible transformations

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO
  HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU
  THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC
  EXAMPLE/
```

Three Views: REs, FSA, RL

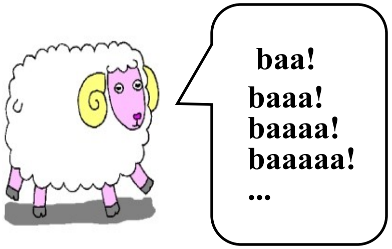
- Three equivalent formal ways to look at what we're up to

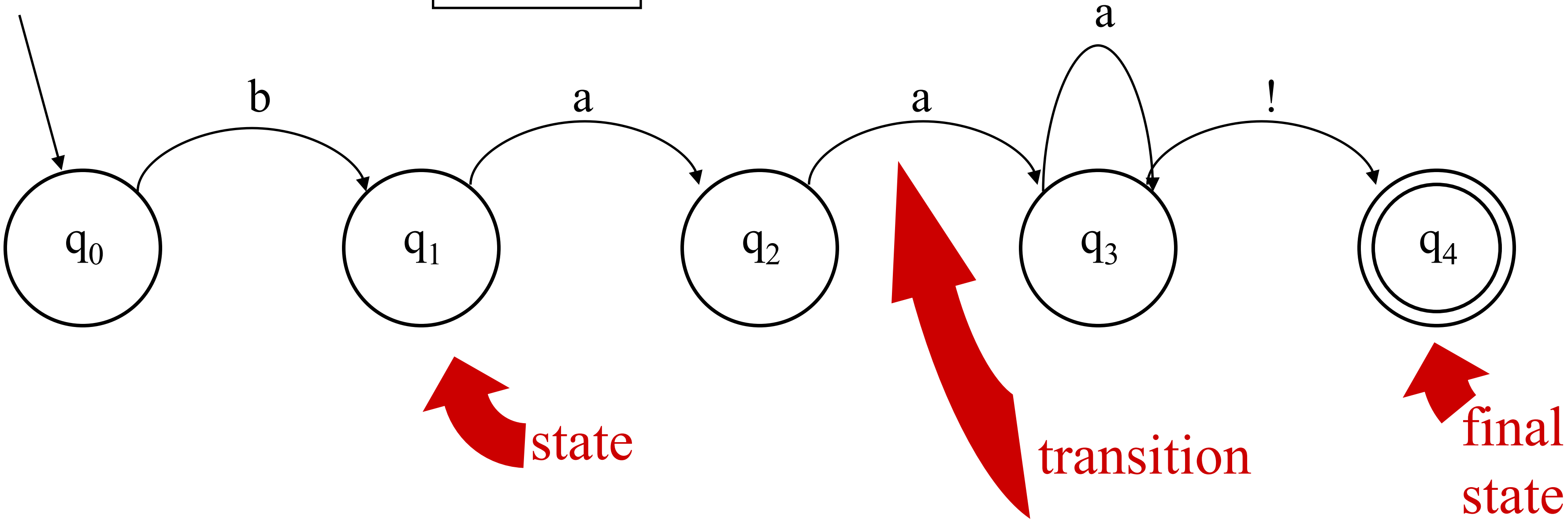


Finite-state automata (machines)

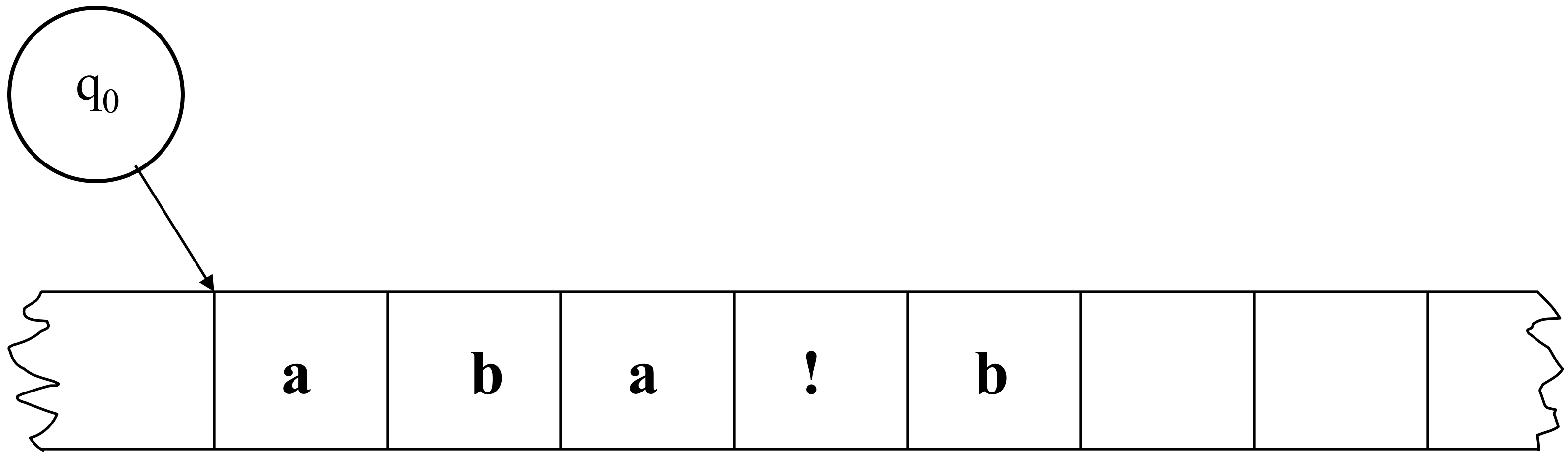
baa!
baaa!
baaaa!
baaaaa!
...

→ baa+!





Input tape



State-transition tables

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

More Formally

- You can specify an FSA by enumerating the following things.
  - The set of states: Q
  - A finite alphabet: Σ
  - A start state
  - A set of accept/final states
  - A transition function that maps Q x Σ to Q

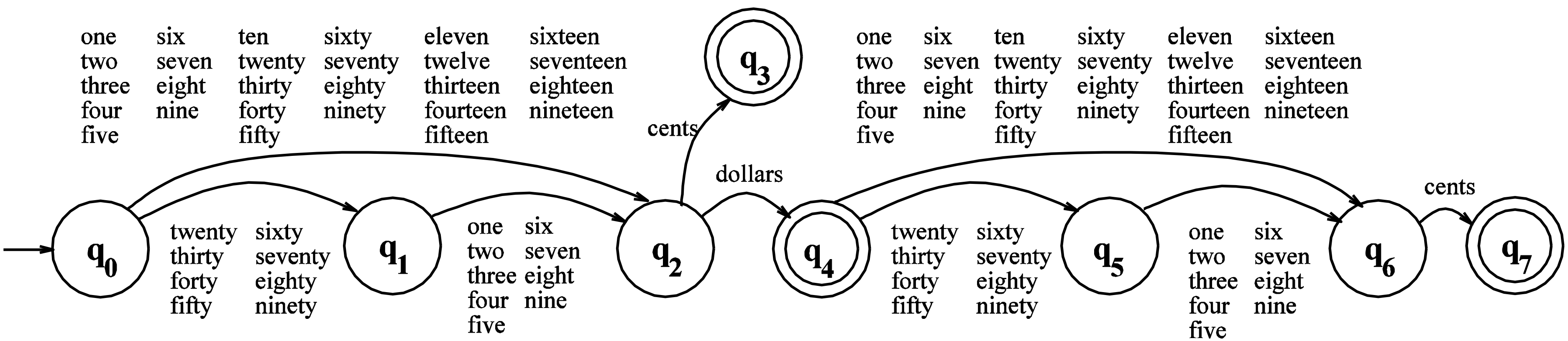
Finite-state automata

- Q: a finite set of N states $q_0, q_1, \ldots q_N$
- Σ: a finite input alphabet of symbols
- q_0: the start state
- F: the set of final states
- $\delta(q,i)$: transition function

Alphabets

- Alphabets means we need a finite set of symbols in the input.
- These symbols can and will stand for bigger objects that can have internal structure.

Dollars and Cents



Recognition

- The process of determining if a string should be accepted by a machine
- The process of determining if a string is in the language we’re defining with the machine
- The process of determining if a regular expression matches a string

Recognition

- in the start state
- Examine the current input (tape)
- Consult the transition table
- Go to the next state and update the tape pointer
- Repeat until you run out of tape

D-RECOGNIZE

```
function D-RECOGNIZE (tape, machine) returns accept or reject
  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached then
      if current-state is an accept state then
        return accept
      else
        return reject
    else if transition-table [current-state, tape[index]] is empty then
      return reject
   else
     current-state ← transition-table [current-state, tape[index]]
     index ← index + 1
end
```

D-Recognize

- Deterministic means that at each point in processing there is always one unique thing to do (no choices)
- D-recognize algorithm is a simple table-driven interpreter
- The algorithm is universal for all unambiguous languages
  - To change the machine, you change the table

Recognition as Search

- You can view this algorithm as a kind of state-space search
- States are pairings of tape positions and state numbers
- Operators are compiled into the table
- Goal state is a pairing with the end of tape position and a final accept state

Generative Formalisms

- Formal Languages are sets of strings composed of symbols from a finite set of symbols
- Finite-state automate define formal languages (without having to enumerate all the strings in the language)
- The term Generative is based on the view that you can run the machine as a generator to get strings from the language

Generative Formalisms

- FSAs can be viewed from two perspectives:
  - Acceptors that can tell you if a string is in the language
  - Generators to produce all and only the strings in the language

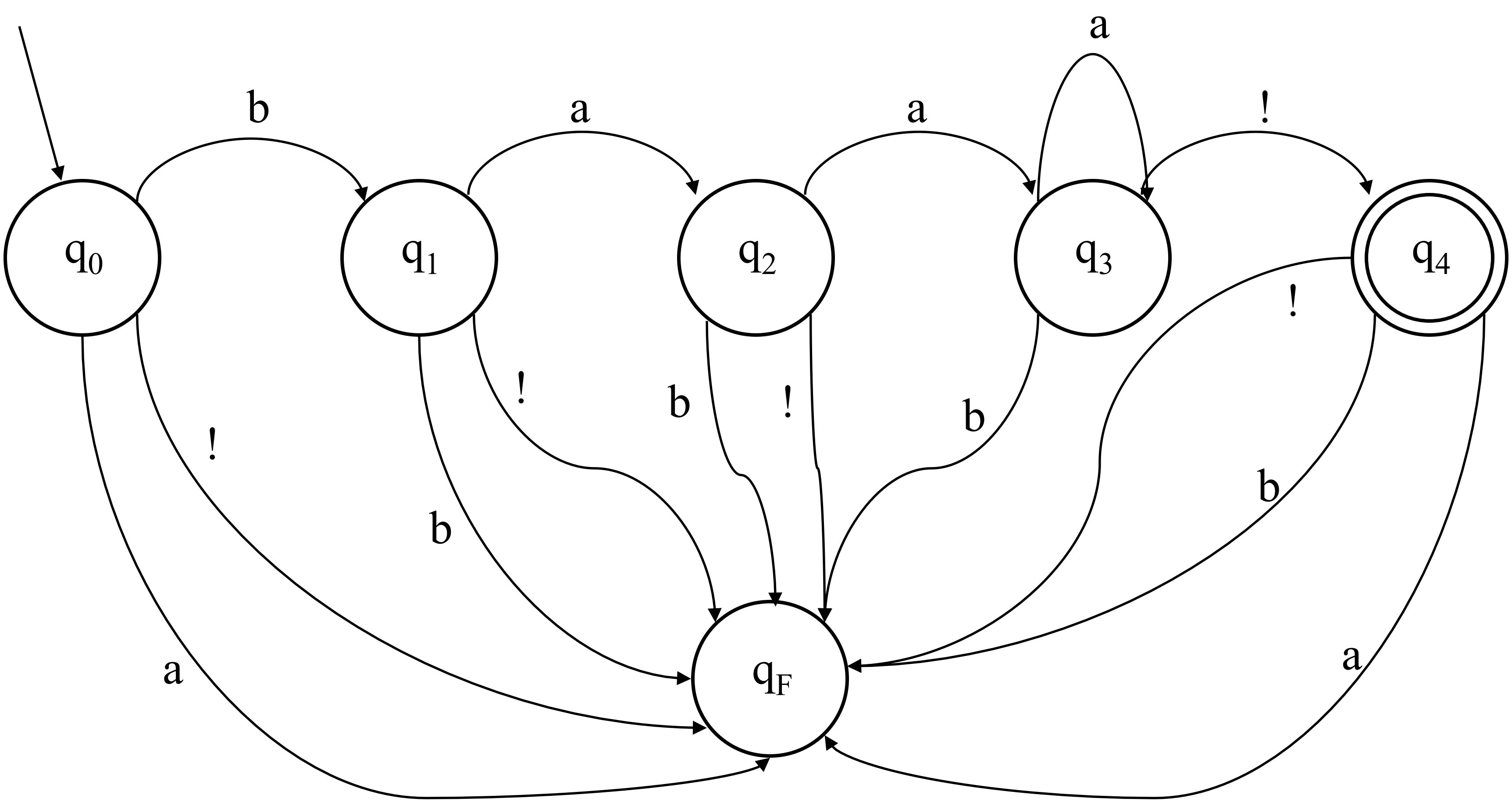
D-RECOGNIZE

```
function D-RECOGNIZE (tape, machine) returns accept or reject
  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached then
      if current-state is an accept state then
        return accept
      else
        return reject
    else if transition-table [current-state, tape[index]] is empty then
      return reject
    else
      current-state ← transition-table [current-state, tape[index]]
      index ← index + 1
end
```

Deterministic Algorithm

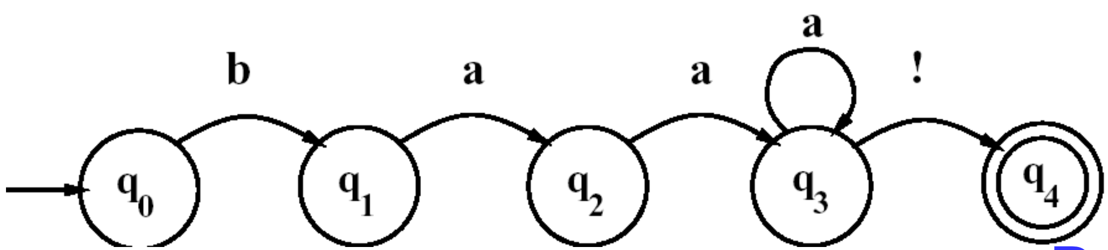
1. Index the tape to the beginning and the machine to the initial state.
2. First check to see if you have any more input
   - If no and you're in a final state, ACCEPT
   - If no and you're in a non-final state reject
3. If you have more input check what state you're in by consulting the transition table. The index of the **Current State** tells you what row in the table to consult. The index on the tape symbol tells you what column to consult in the table. Loop through until no more input then go back to 2.

Adding a failing state

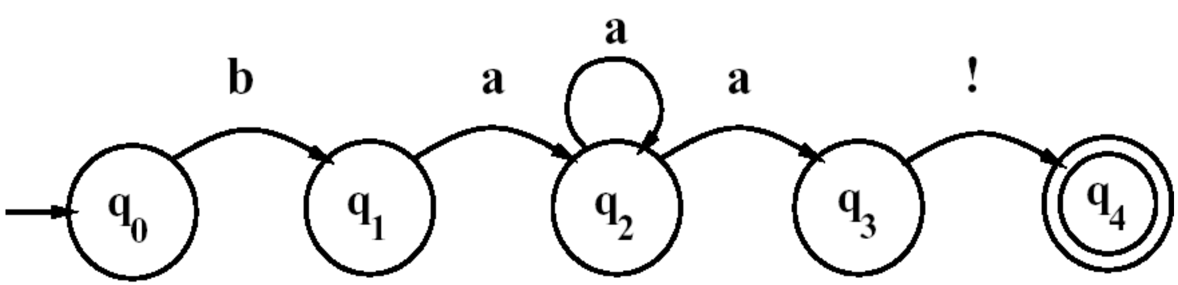


Languages and automata

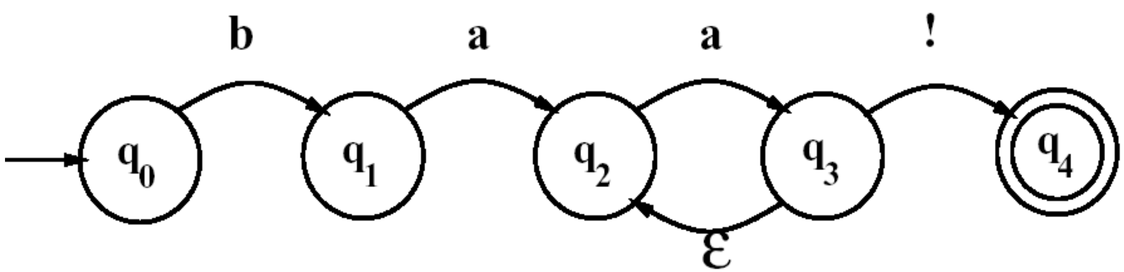
- Formal languages: regular languages, non-regular languages
- deterministic vs. non-deterministic FSAs
- Epsilon (ε) transitions
  - ε is the empty string & Ø is the empty set (empty regular language)



Deterministic



Non- Deterministic



Non- Deterministic

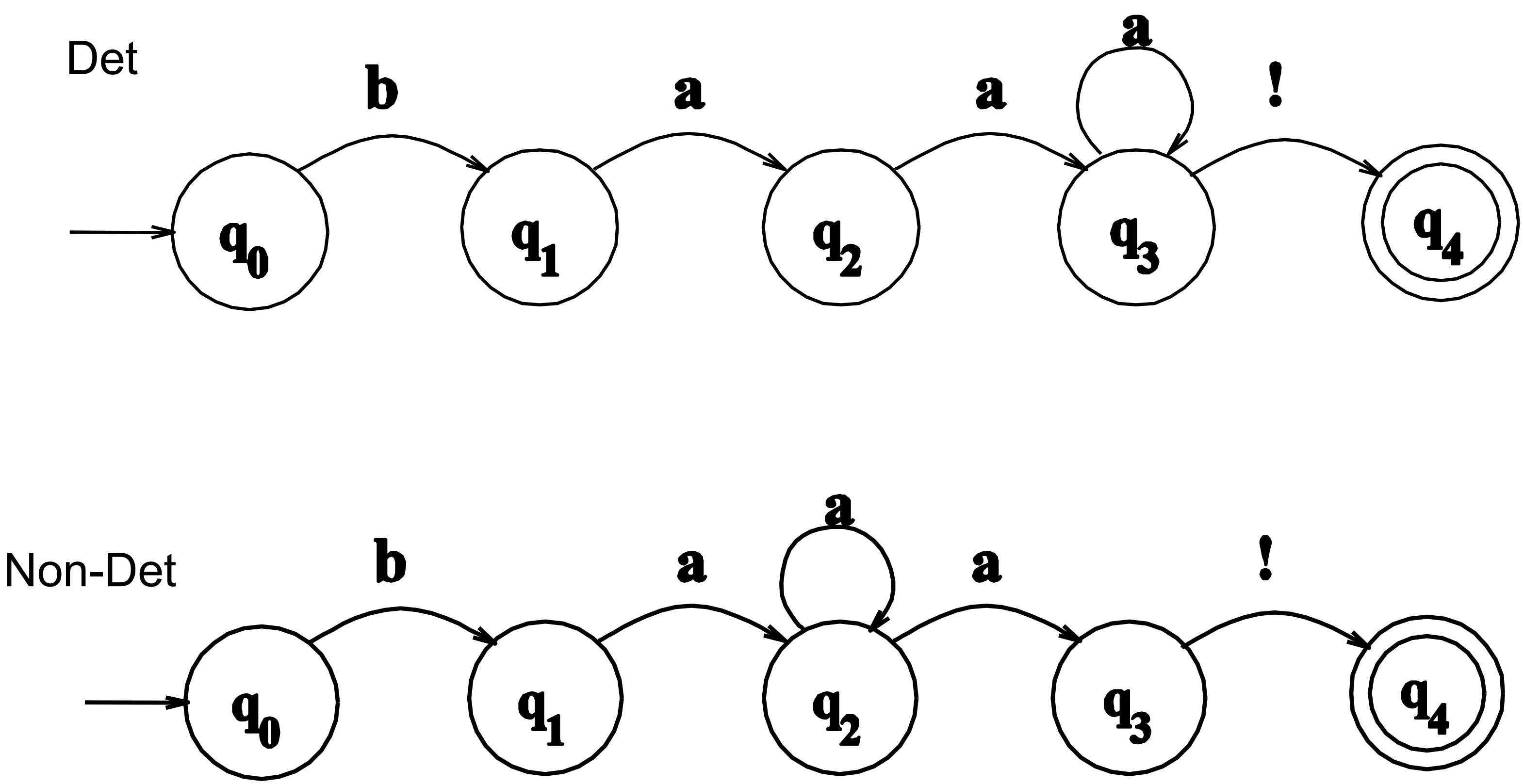
Using NFSAs to accept strings

- Backup: add markers at choice points, then possibly revisit unexplored markers
- Look-ahead: look ahead in input
- Parallelism: look at alternatives in parallel

Using NFSAs

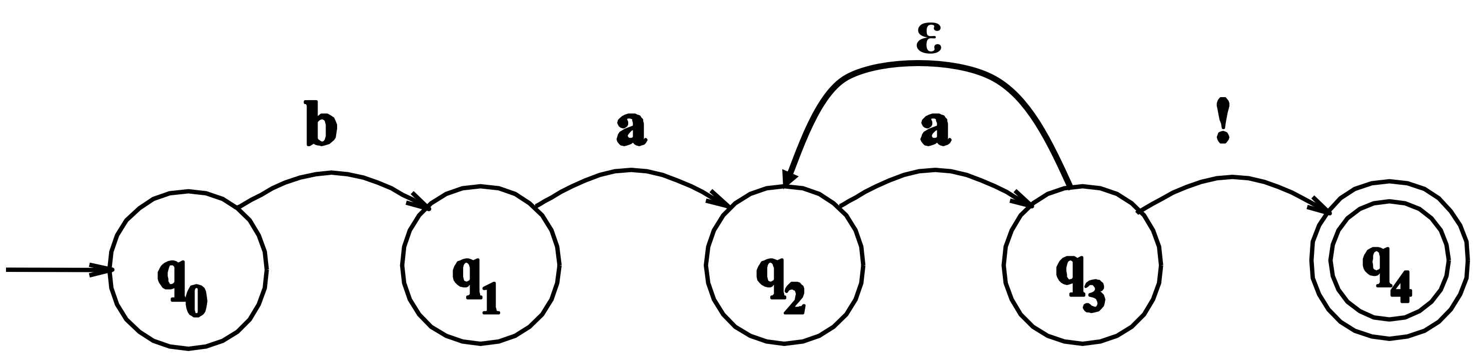
	Input			
State	b	a	!	ε
0	1	Ø	Ø	Ø
1	Ø	2	Ø	Ø
2	Ø	2,3	Ø	Ø
3	Ø	Ø	4	Ø
④	Ø	Ø	Ø	Ø

Non-Determinism



Non-Determinism cont.

- Another technique
  - Epsilon transitions
    - these transitions do not examine or advance the tape during recognition



Equivalence

- Non-deterministic machines can be converted to deterministic
- They have the same power; non-deterministic machines are not more powerful than deterministic ones
- It also means that one way to do recognition with a non-deterministic machine is to turn it into a deterministic one

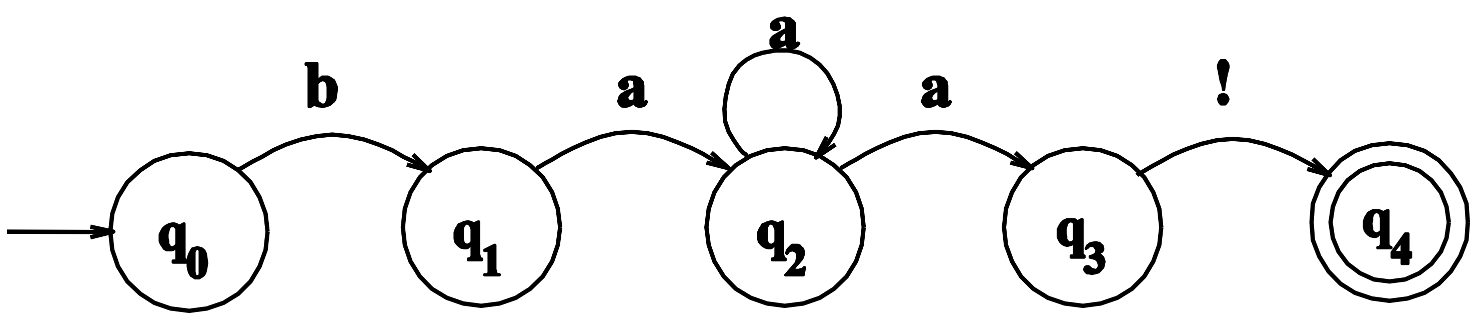
Non-Deterministic Recognition

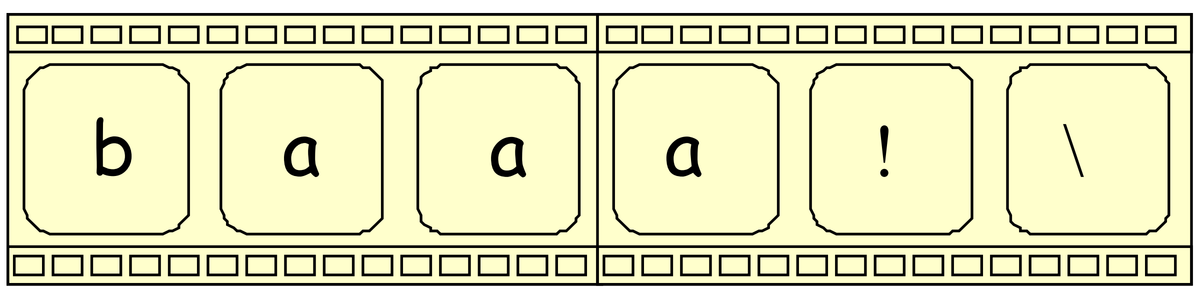
- In a ND FSA there exists at least one path through the machine for a string that is in the language defined by the machine
- But not all paths directed through the machine for an accept string lead to an accept state
- No paths through the machine lead to an accept state for a string not in the language

Non-Deterministic Recognition

- Success in a non-deterministic recognition occurs when a path is found through the machine that ends in an accept
- Failure occurs when none of the possible paths lead to an accept state

Example

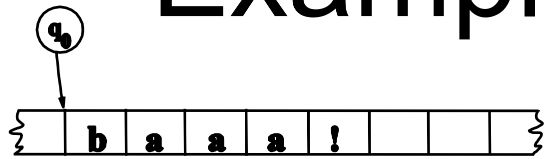


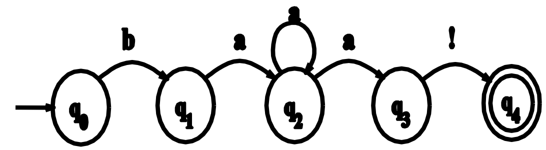


q_0 q_1 q_2 q_2 q_3 q_4

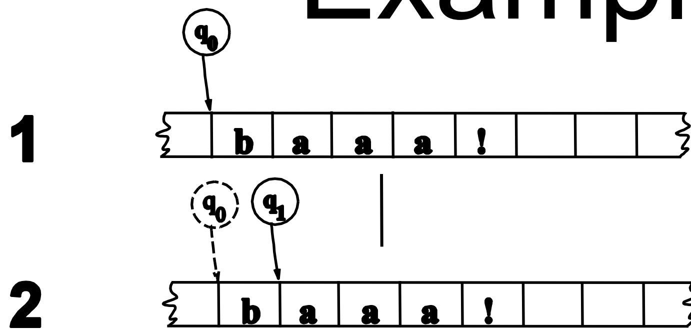
Example

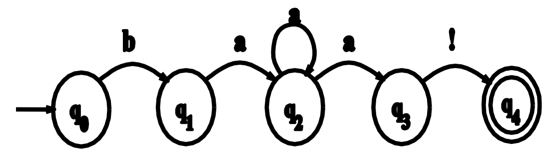
1



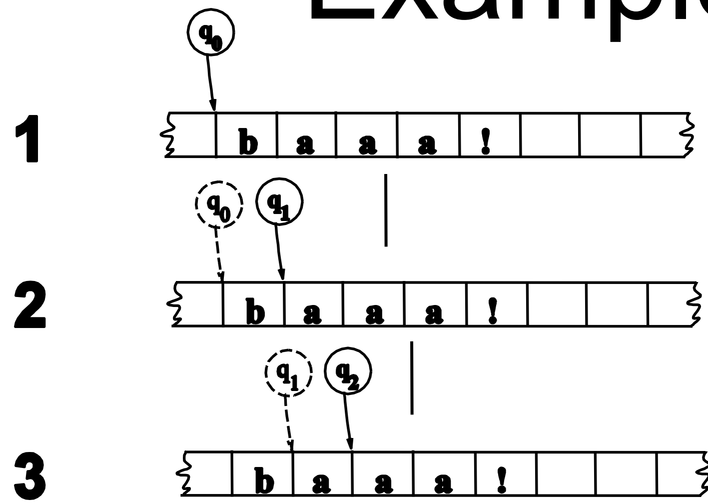


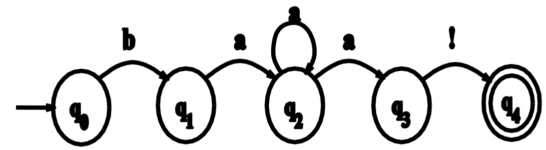
Example



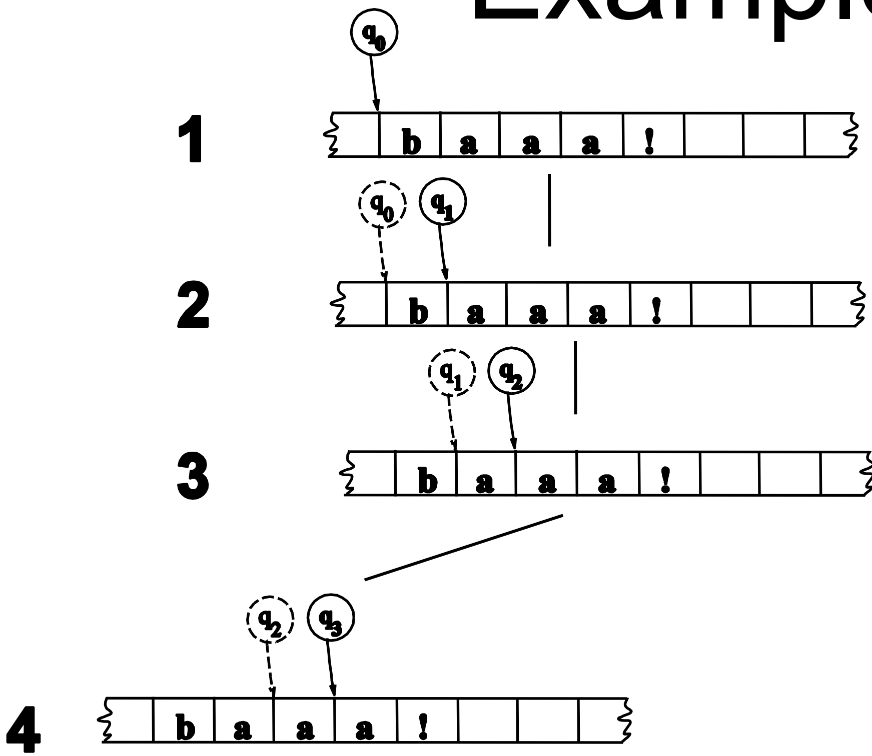


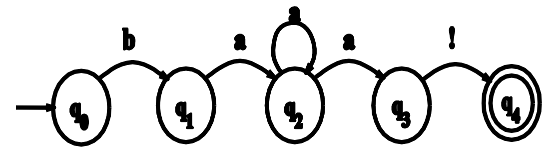
Example



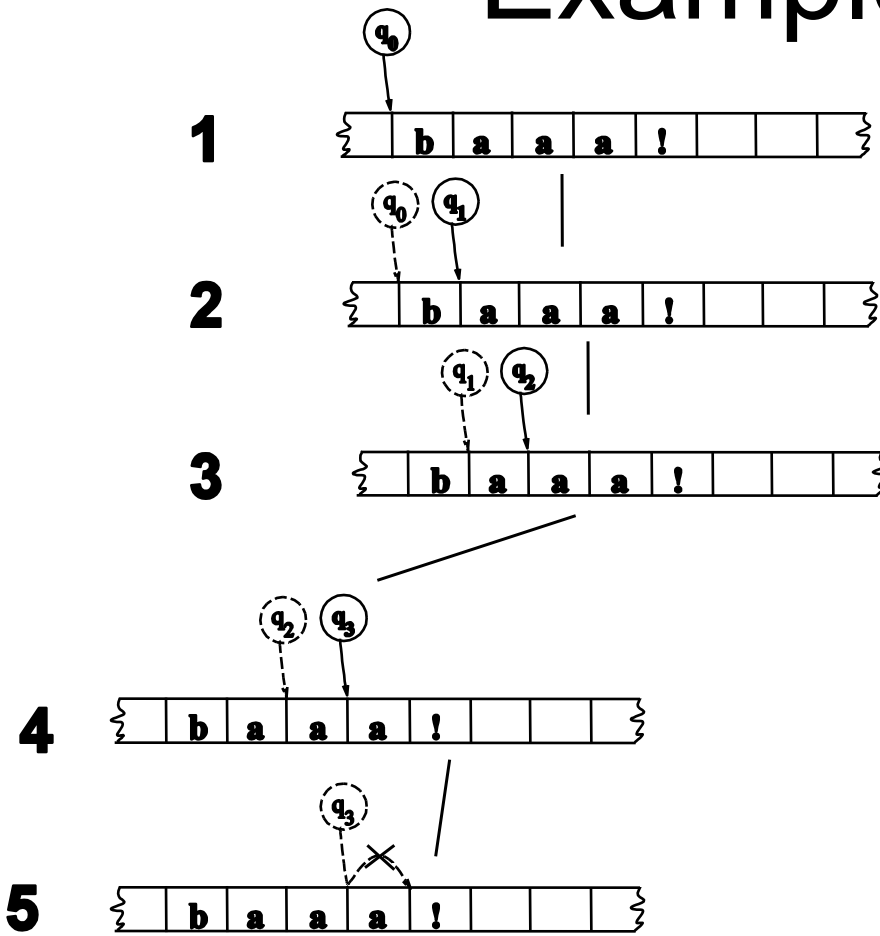


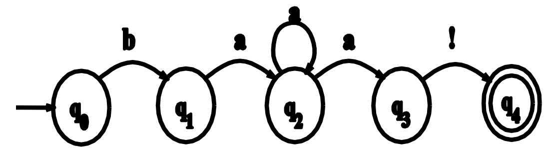
Example



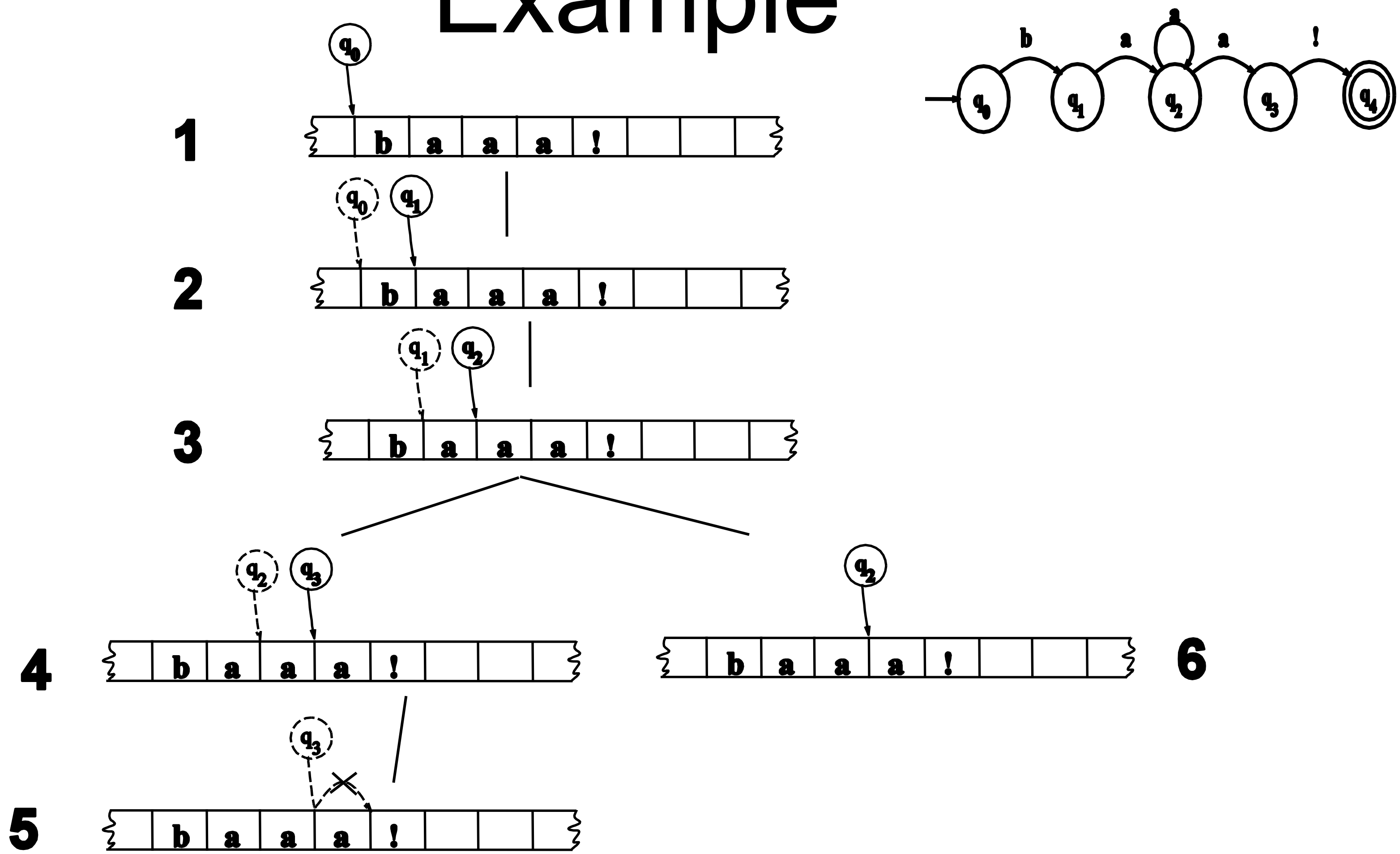


Example

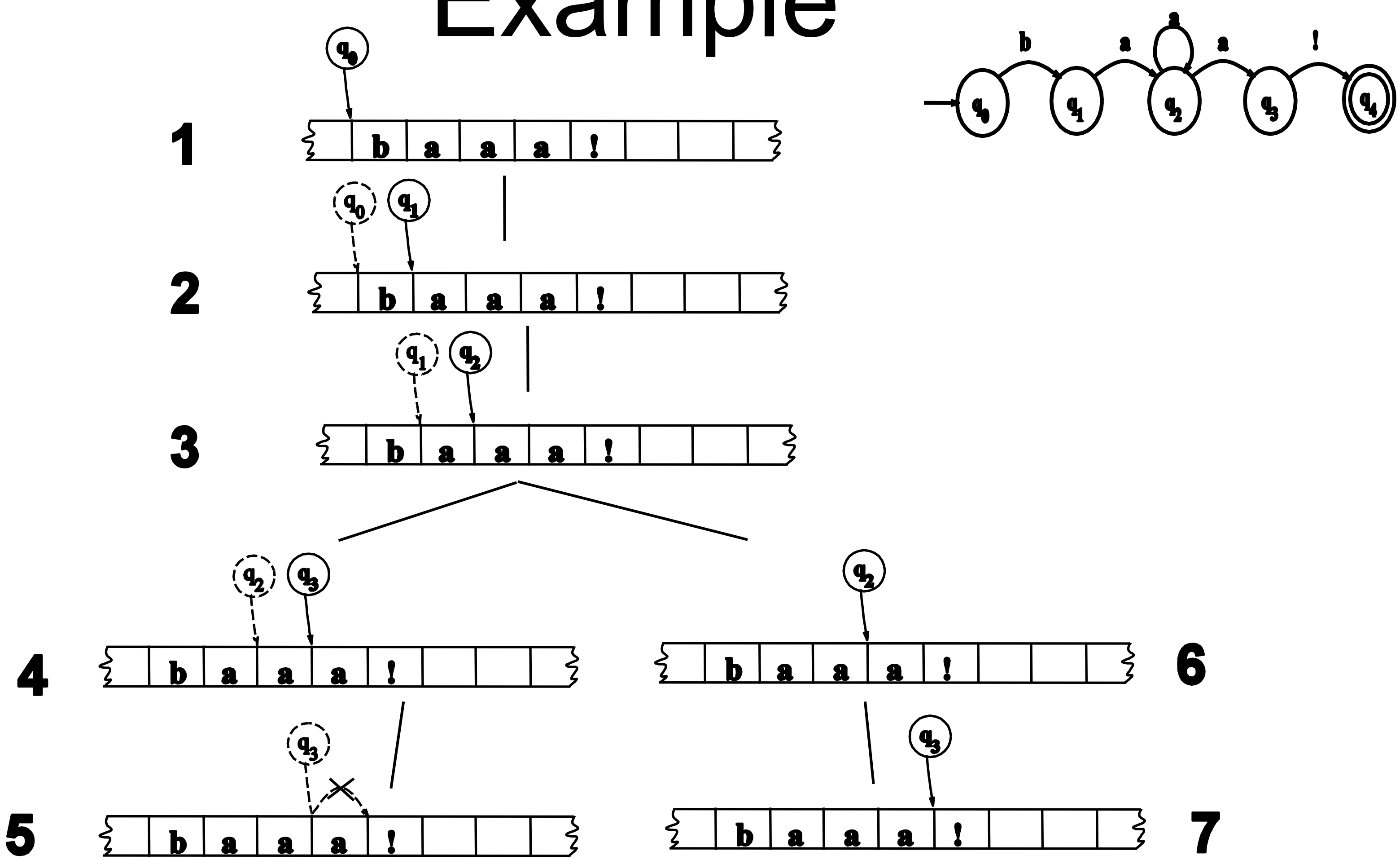




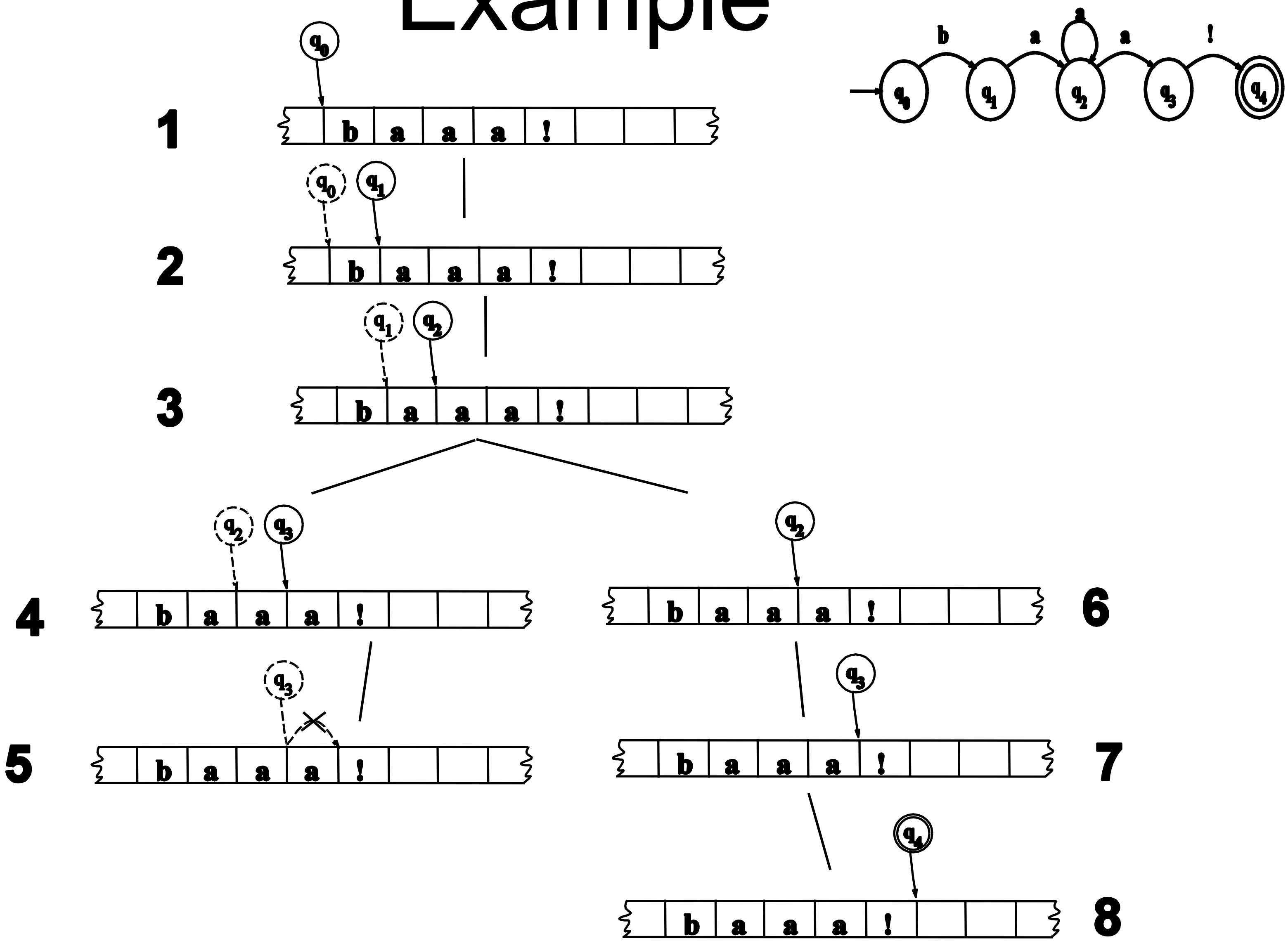
Example



Example



Example



States in Search Space

- States in the search space are pairings of tape positions and states in the machine
- By keeping track of as yet unexplored states, a recognizer can systematically explore all the paths through the machine given an input

Components of ND Automaton

Search State: records the choice points by storing state, input pairs so you know what state you were at and what input you had read when the derivation branched.

Agenda: At each point of nondeterminism the algorithm postpone pursuing some choices (paths) in favor of others. The agenda records what these choices are as they are encountered.

Since this is **non-deterministic**, we have to allow the state to transition to multiple points (a **list of destination nodes**).

Non Deterministic Algorithm

1: Can you accept the string given input and state
2: If not , check the agenda and given the current state and the input then generate a new set of possible search states based on the state you are in and new input. Explore these states.
3: If not, see if there are alternative search states waiting to be explored on the agenda.
If either (2) or (3) end up, the states they lead you to become the current search state.
Even if one path doesn't succeed always need to check the agenda because you may come to (Final state, 0-input pair) on another path.

Search in NFSA

- Depth first Search
  - Last in First Out (LIFO)
  - States arranged in a STACK
- Breadth first Search
  - First in first out (FIFO)
  - States organized in a queue

When to choose what?

- **Depth first search** is optimal when one alternative is highly favored because in most cases, you will never get to the less favored alternatives.
- **Breadth first search** is optimal when can’t predict which alternative likely to work out. You will do extra work by computing paths that won’t lead to final output, but when error is detected at one path, don’t have to back up to get to other paths. Can just proceed with next step.
- Unfortunately, often can’t tell which will save the most work.

Infinite Search

- If we're not careful such searches can go into an infinite loop.

Why to use Non-determinism

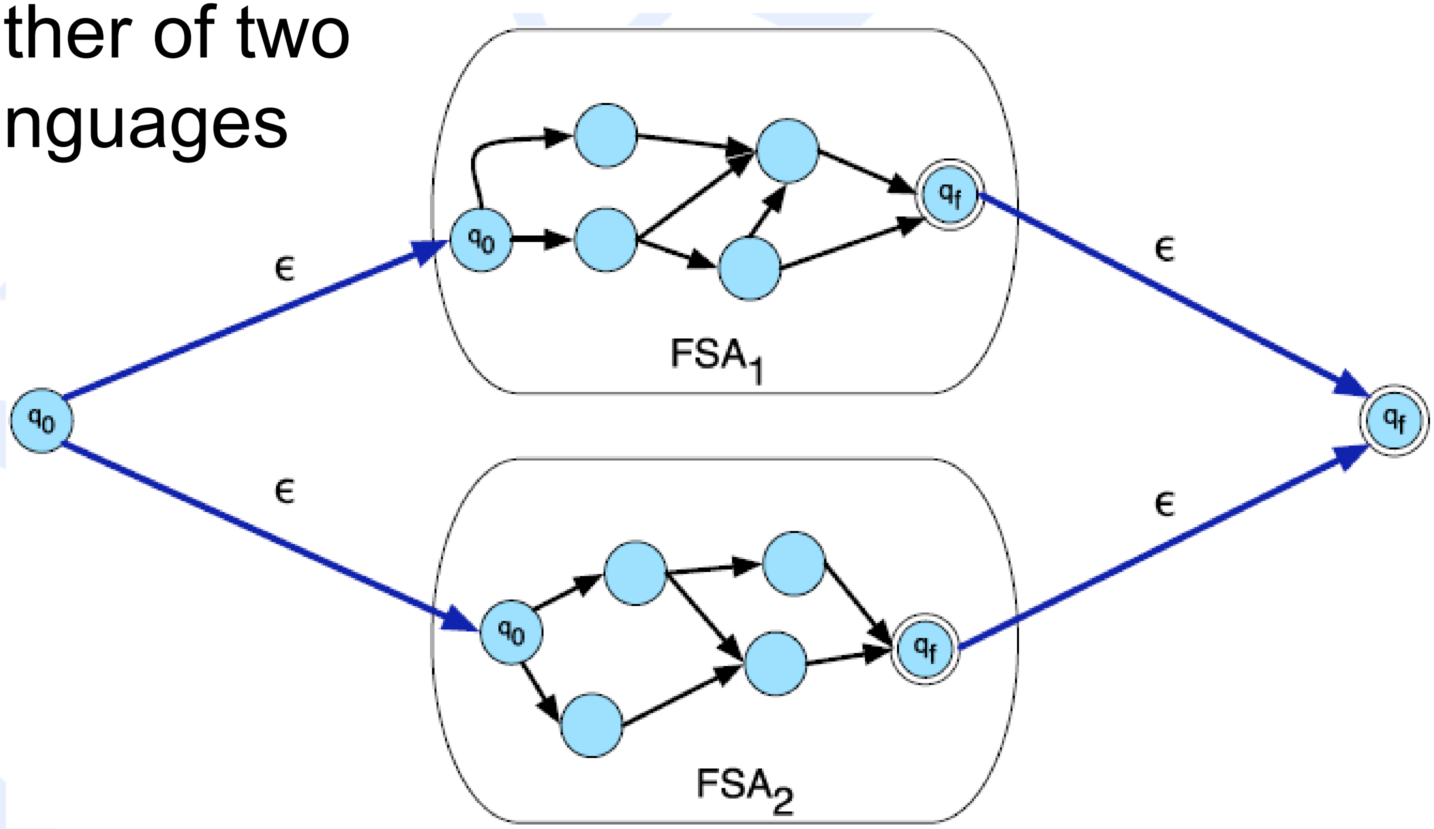
- Non-determinism doesn't get us more formal power and it causes headaches so why to use it?
  - More natural solutions
  - Deterministic Machines are too big

Compositional Machines

- Formal languages are sets of strings
- We can talk about various set operations (intersection, union, concatenation)

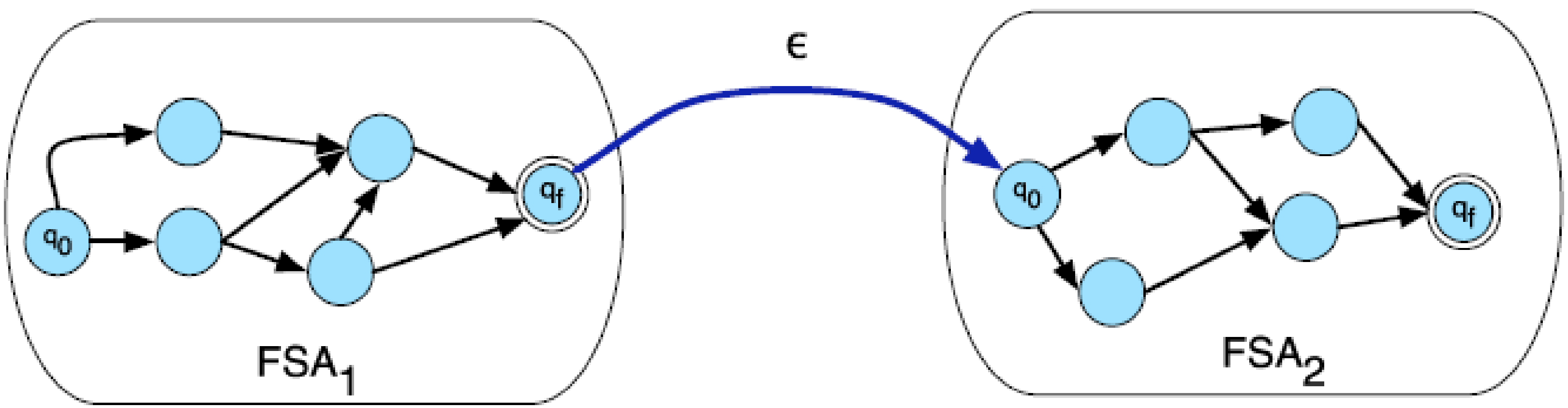
Union

- Accept a string in either of two languages



Concatenation

- Accept a string consisting of a string from language L1 followed by a string from language L2.



Negation

- Construct a machine M2 to accept all strings not accepted by machine M1 and reject all the strings accepted by M1
  - Invert all the accept and not accept states in M1
- Does that work for non-deterministic machines?

Intersection

- Accept a string that is in both of two specified languages
- An indirect construction…
    - A^B = ~(~A or ~B)

Thank you

السلام عليكم ورحمة الله