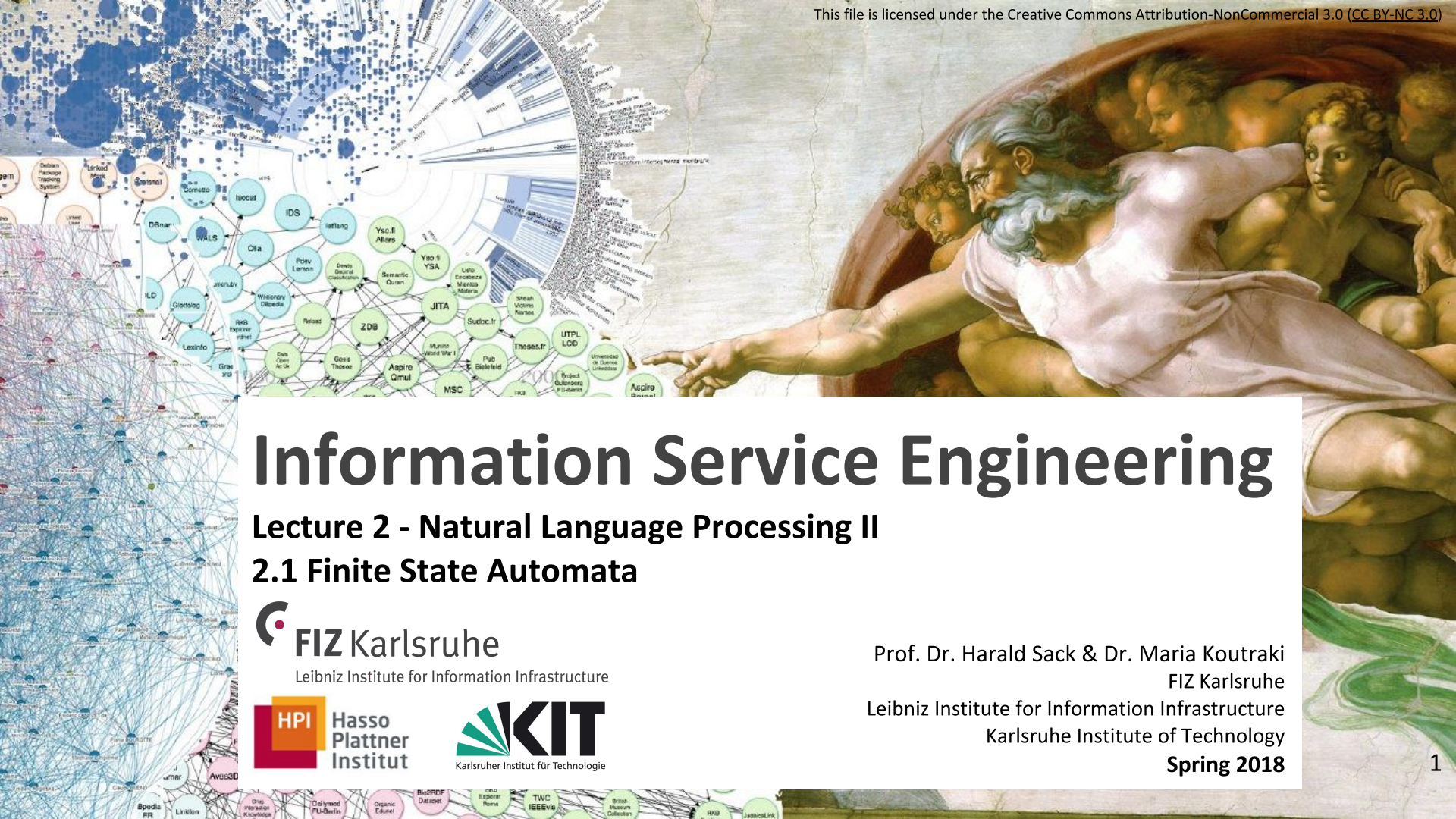




Information Service Engineering

Lecture 2 - Natural Language Processing II

2.1 Finite State Automata



FIZ Karlsruhe

Leibniz Institute for Information Infrastructure



Hasso
Plattner
Institut



Karlsruher Institut für Technologie

Prof. Dr. Harald Sack & Dr. Maria Koutraki
FIZ Karlsruhe

Leibniz Institute for Information Infrastructure
Karlsruhe Institute of Technology
Spring 2018

Information Service Engineering

Lecture 2: Natural Language Processing II

2.1 Finite State Automata

2.2 Finite State Transducers

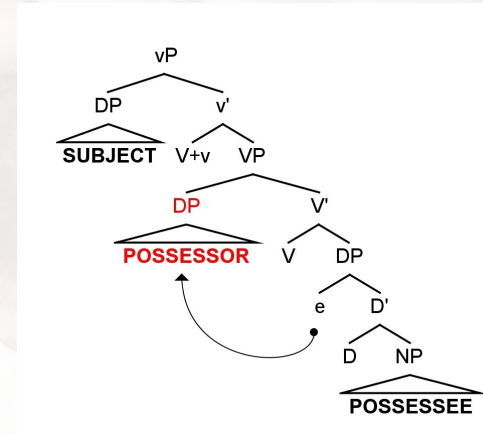
2.3 Language Model and N-Grams

2.4 Language Model and N-Grams II

2.5 Language Model and N-Grams III

2.6 Part-of-Speech Tagging

2.7 Part-of-Speech Tagging II



Regular Expressions and Finite State Automata (FSA)

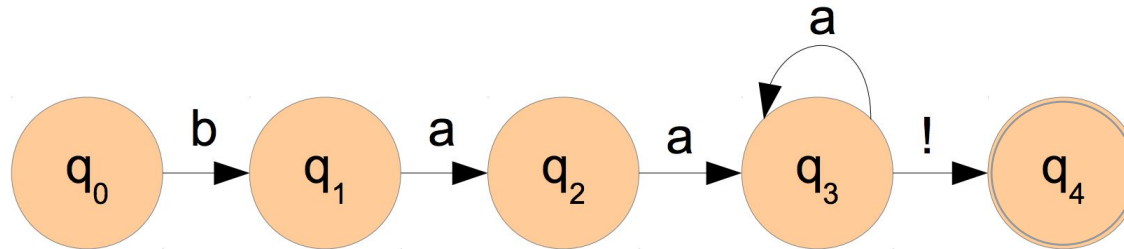
Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

- **Regular Expressions**
 - are one way to characterize a **Regular Language**
as e.g. `/baa+!/?`
- **Regular Languages** can be characterized via
 - Regular expressions
 - **Finite State Automata**
 - Regular grammars

Finite State Automaton

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

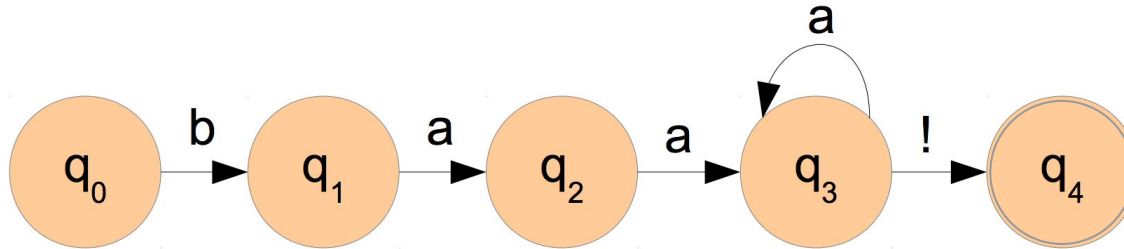
- A **Finite State Automaton** is an abstract model of a computer which
 - **reads an input string,**
 - **and changes its internal state**
depending on the current input symbol.
- An FSA can either **accept** or **reject** the input string.
- Every automaton defines a **language**, i.e. the set of strings it accepts



Finite State Automaton

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

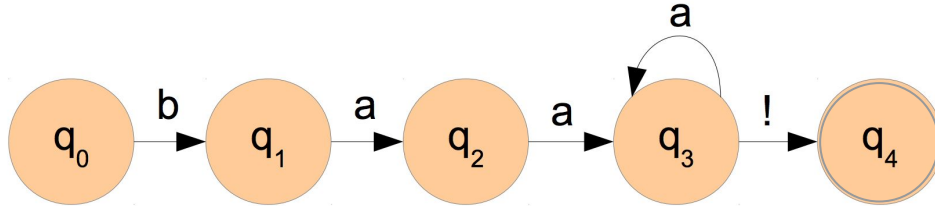
- **Finite State Automata** are composed of
 - Vertices (nodes)
 - Arcs (links)



- What words (strings) can be recognized by this FSA example?
 - **baa!** / **baaa!** / **baaaa!** / **baaaaa!** / ..
- Matching RE?
 - **/baa+!/?**

Finite State Automaton

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata



- **Strings accepted:**
 - baa!
 - baaaaa!
- **Strings not accepted:**
 - abc
 - babb
 - !bcd

State Transition Table

	Input		
State	a	b	!
q ₀	∅	q ₁	∅
q ₁	q ₂	∅	∅
q ₂	q ₃	∅	∅
q ₃	q ₃	∅	q ₄
q ₄	∅	∅	∅

Formalization of FSAs

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

- **FSA** $A = (Q, \Sigma, q_0, F, \delta(q,i))$, with
- **Q**: finite set $\{q_0, q_1, q_2, \dots, q_{N-1}\}$ of N states
- **Σ** : finite input alphabet of symbols
- **q_0** : the designated start state
- **F**: the set of final states, $F \subseteq Q$
- **$\delta(q,i)$** : the transition function
 $\delta: Q \times \Sigma \rightarrow Q$,
 $\delta(q,i) = q'$ for $q, q' \in Q, i \in \Sigma$

Example: Formalization of “Sheeptalk”

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

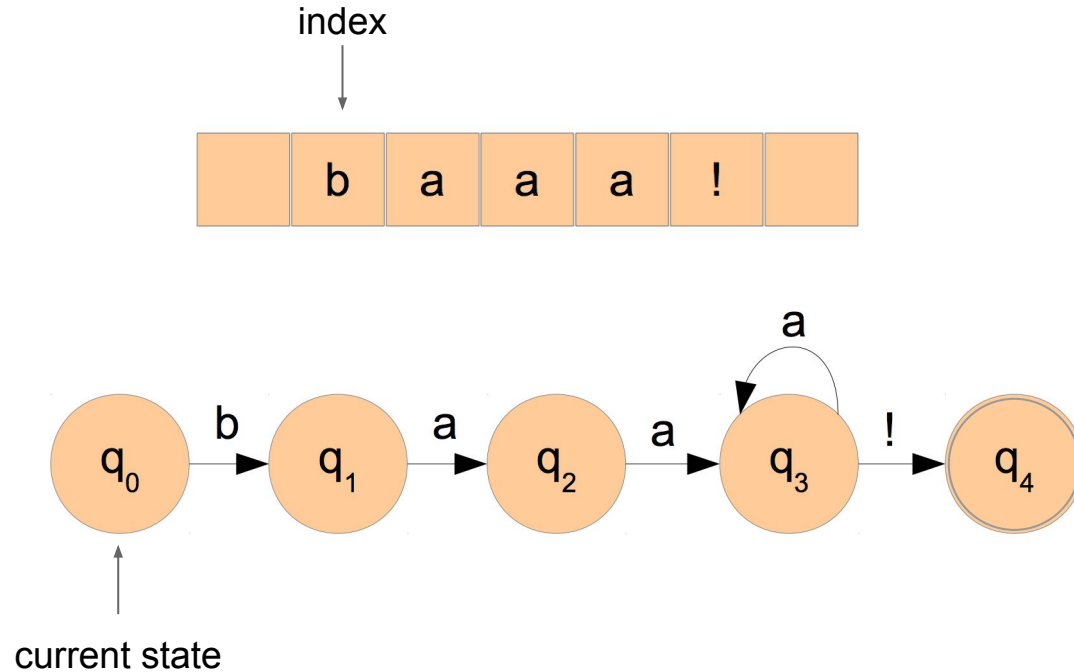
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b, !\}$
- q_0 : the start state
- $F = \{q_4\}$
- $\delta(q, i)$: defined by state transition table

State Transition Table

	Input		
State	a	b	!
q_0	$\emptyset$	q_1	$\emptyset$
q_1	q_2	$\emptyset$	$\emptyset$
q_2	q_3	$\emptyset$	$\emptyset$
q_3	q_3	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$	$\emptyset$

D-RECOGNIZE Algorithm (step 1)

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

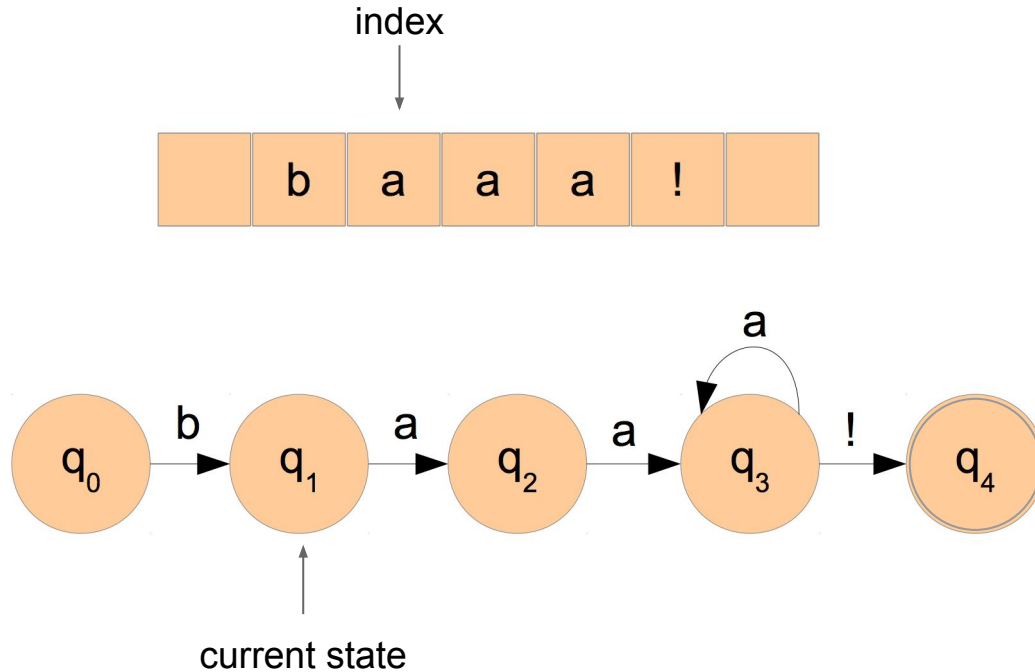


State Transition Table

	Input		
State	a	b	!
q_0	$\emptyset$	q_1	$\emptyset$
q_1	q_2	$\emptyset$	$\emptyset$
q_2	q_3	$\emptyset$	$\emptyset$
q_3	q_3	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$	$\emptyset$

D-RECOGNIZE Algorithm (step 2)

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

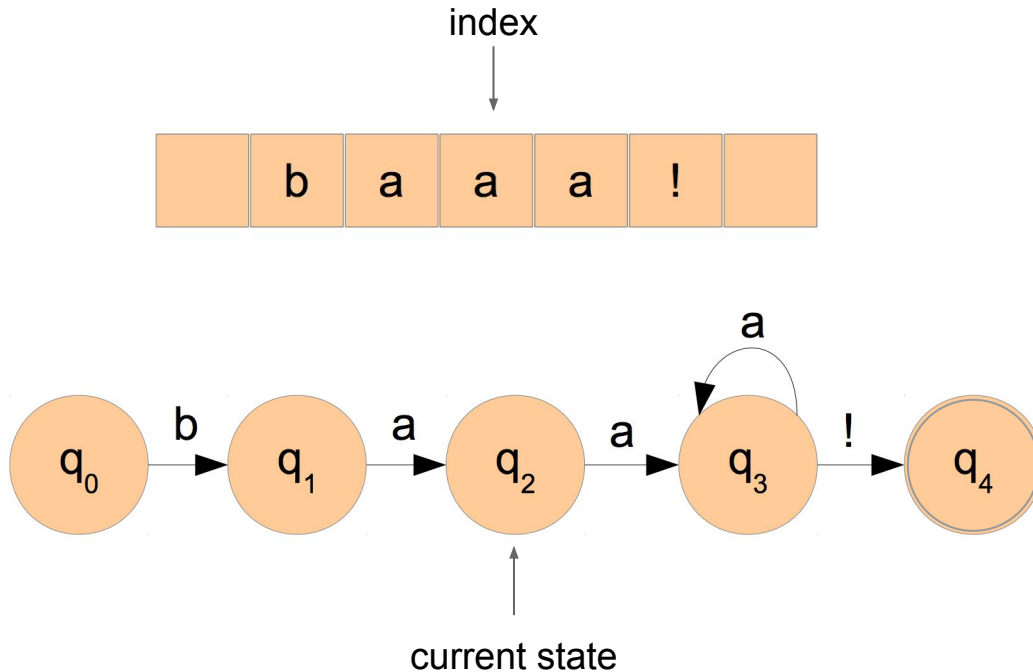


State Transition Table

	Input		
State	a	b	!
q_0	$\emptyset$	q_1	$\emptyset$
q_1	q_2	$\emptyset$	$\emptyset$
q_2	q_3	$\emptyset$	$\emptyset$
q_3	q_3	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$	$\emptyset$

D-RECOGNIZE Algorithm (step 3)

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

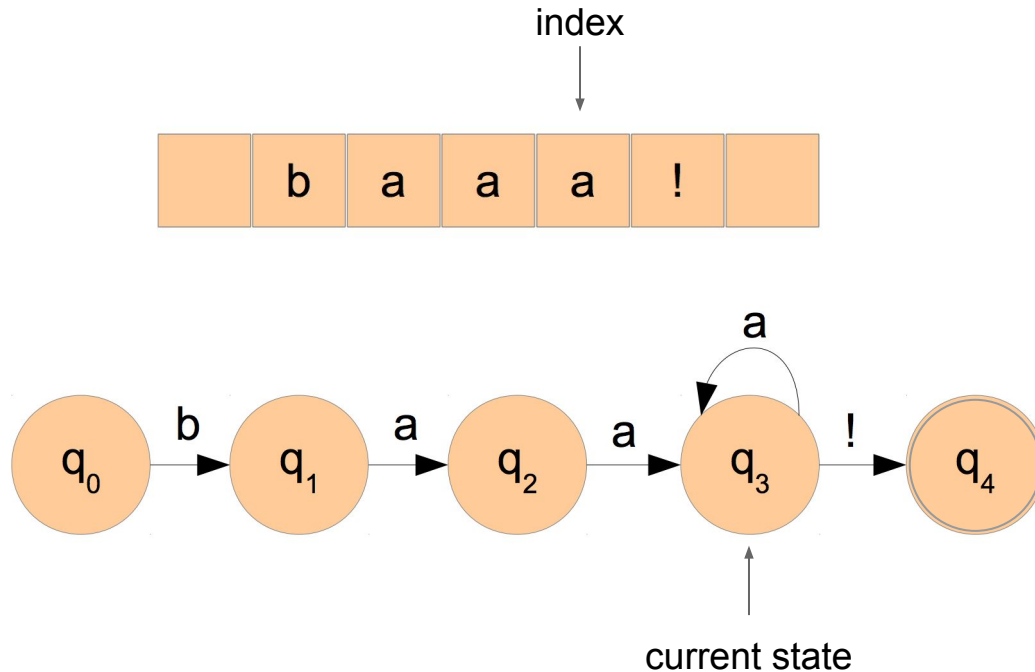


State Transition Table

	Input		
State	a	b	!
q_0	$\emptyset$	q_1	$\emptyset$
q_1	q_2	$\emptyset$	$\emptyset$
q_2	q_3	$\emptyset$	$\emptyset$
q_3	q_3	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$	$\emptyset$

D-RECOGNIZE Algorithm (step 4)

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

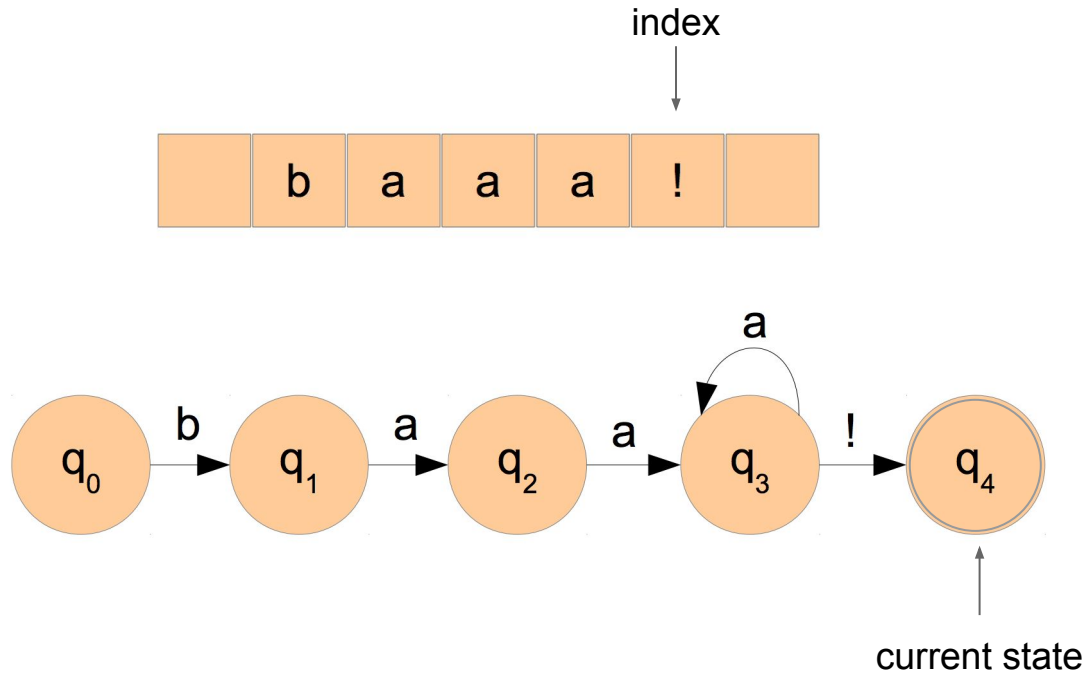


State Transition Table

	Input		
State	a	b	!
q ₀	∅	q ₁	∅
q ₁	q ₂	∅	∅
q ₂	q ₃	∅	∅
q ₃	q ₃	∅	q ₄
q ₄	∅	∅	∅

D-RECOGNIZE Algorithm (step 5)

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata



State Transition Table

	Input		
State	a	b	!
q ₀	∅	q ₁	∅
q ₁	q ₂	∅	∅
q ₂	q ₃	∅	∅
q ₃	q ₃	∅	q ₄
q ₄	∅	∅	∅

- q₄ is the final state, string is accepted.

- A **model** that can both **generate** and **recognize** all and only the strings given by its definition.
- An automaton can describe an infinite set with a closed form.
- “Sheeptalk” model “m”:
 - $\Sigma = \{b, a, !\}$ - Alphabet
 - $L(m)$ = formal language characterized by „m“
 - $L(m) = \{baa!, baaa!, baaaa!, \dots\}$
 - Usually it holds that $L \subset \Sigma^*$
 - The **Kleene closure** Σ^* (‘sigma star’) is the (infinite) set of all strings that can be formed from Σ .

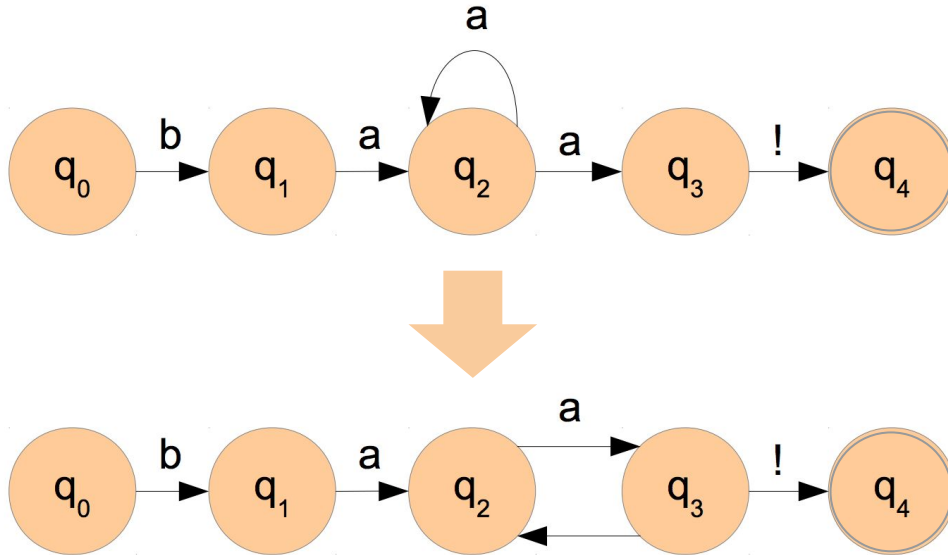
Formal Language vs. Natural Language

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

- A **formal language** is not a **natural language**
- But we can use formal languages to model parts of natural languages,
 - such as e.g. **phonology**, **morphology** or **syntax**

Non-Deterministic FSA

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata



- $\delta(q,i)$: the transition function for NFAs
 $\delta: Q \times \Sigma_{\epsilon} \rightarrow 2^Q$, $\Sigma_{\epsilon} = \Sigma \cup \{\epsilon\}$, $\Sigma \cap \{\epsilon\} = \emptyset$

$$\delta(q,i) = Q' \text{ for } q \in Q, Q' \subseteq Q, i \in \Sigma_{\epsilon}$$

State Transition Table

	Input			
State	a	b	!	ϵ
q_0	$\emptyset$	q_1	$\emptyset$	$\emptyset$
q_1	q_2	$\emptyset$	$\emptyset$	$\emptyset$
q_2	q_3	$\emptyset$	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	q_4	q_2
q_4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Finite State Morphological Parser

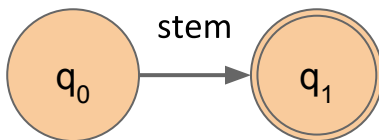
Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

- To construct a morphological parser, we need:
 - **Lexicon**: list of **stems** and **affixes** + **properties** (noun, verb, etc.)
 - **Morphotactics**: model of **morpheme ordering**,
e.g. *plural affix -s follows noun*
 - **Orthographic rules**: spelling rules for morpheme combinations,
e.g. **y** → **ie** that changes **city** + **-s** into **cities**

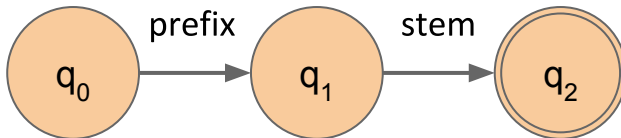
FSA for Morphology

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

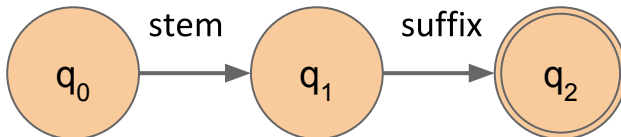
- grace:



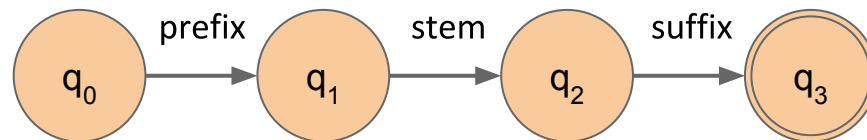
- dis-grace:



- grace-ful:



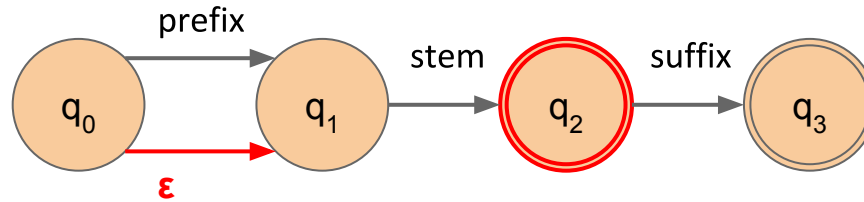
- dis-grace-ful:



Merging Automata

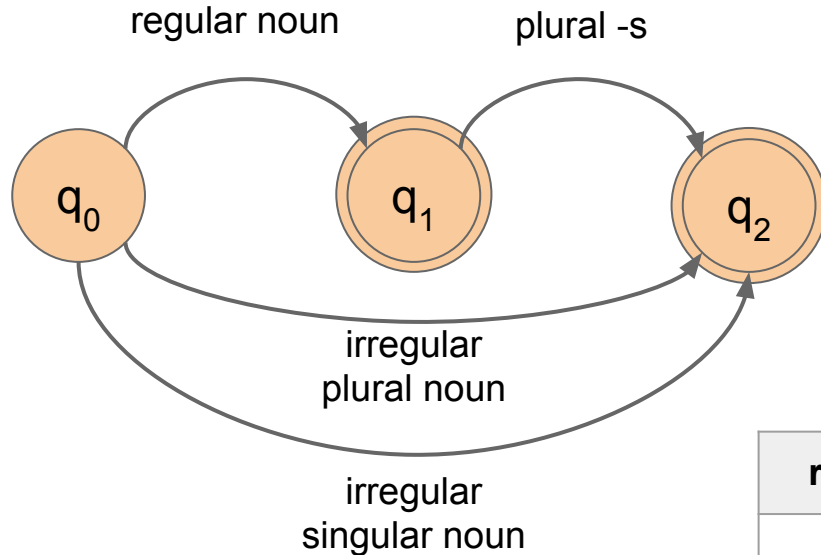
Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

- grace,
dis-grace,
grace-ful,
dis-grace-ful



Simple FSA for English Nominal Inflection

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata

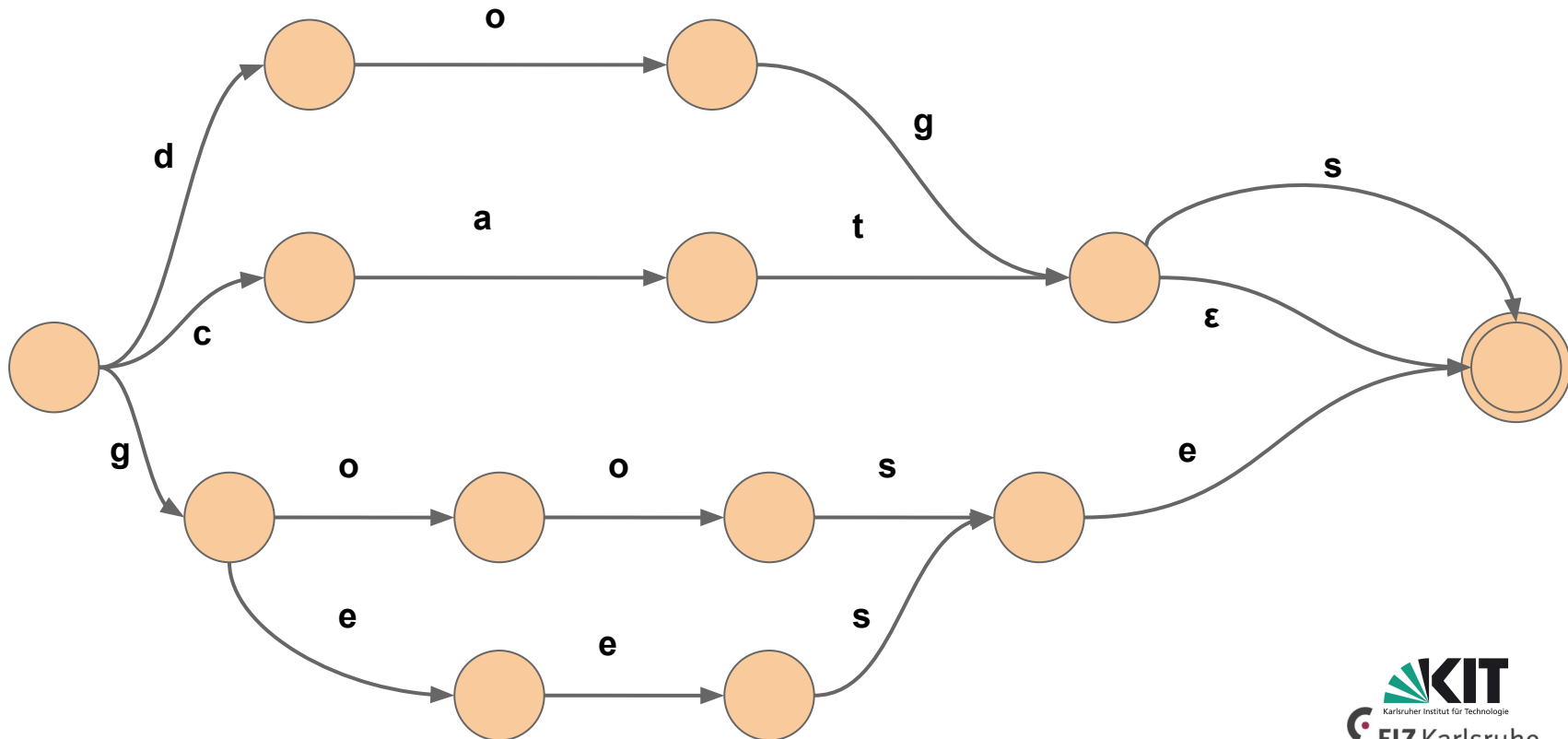


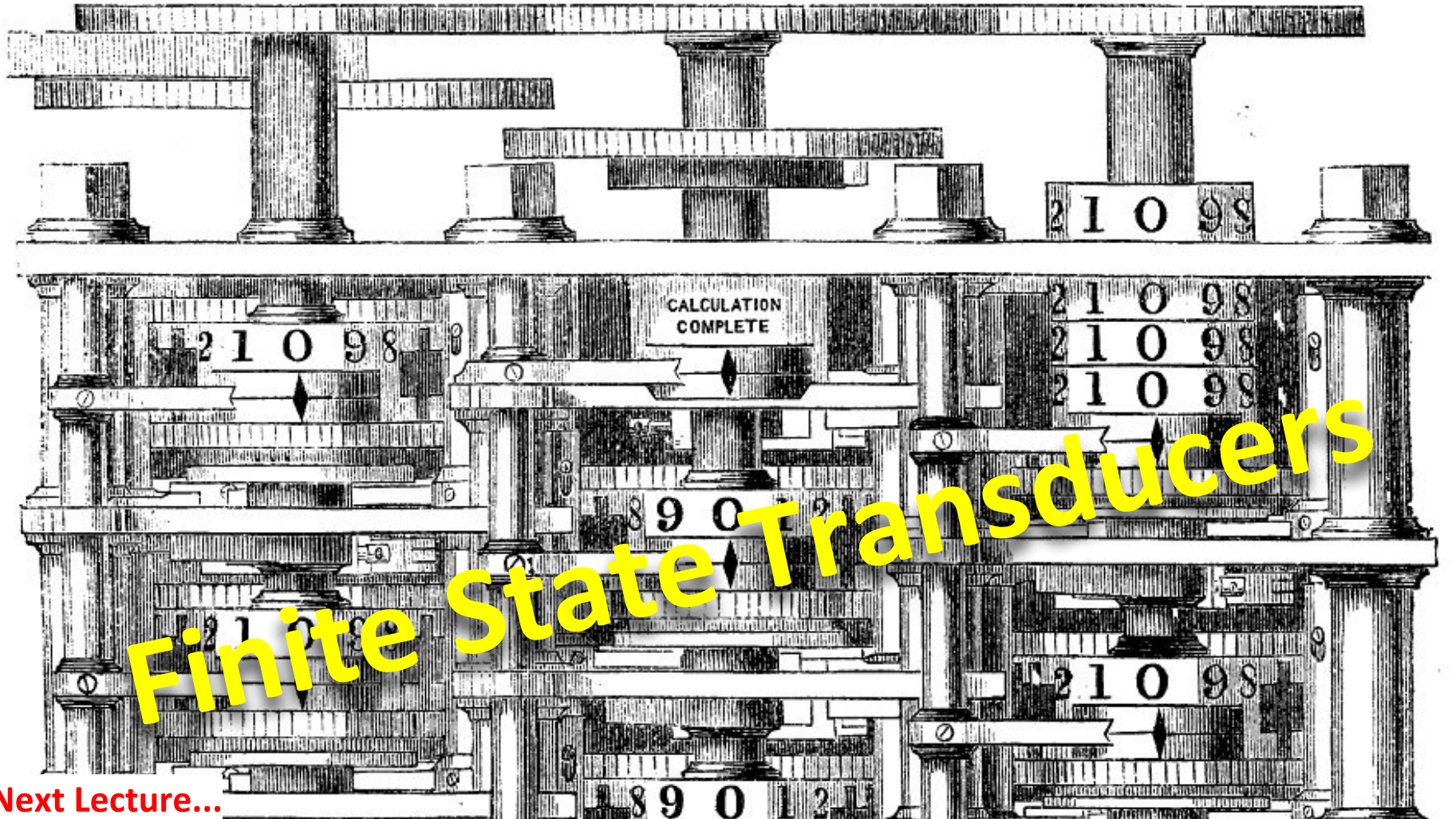
- Some irregular words require stem changes, e.g. goose - geese

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s
cat	sheep	sheep	
aardvark	mice	mouse	

Expanded FSA for a few English Nouns

Information Service Engineering - Lecture 2 Natural Language Processing II - 2.1 Finite State Automata





Finite State Transducers

Next Lecture...

Picture References:

- P.22: Portion of Babbage's difference engine (1864),
https://commons.wikimedia.org/wiki/File:Babbage_difference_engine_drawing.gif