Natural Language Processing
SoSe 2016

IT Systems Engineering | Universität Potsdam

Regular Expressions, Automata, Morphology and Transducers

Dr. Mariana Neves

April 25th, 2016

Outline

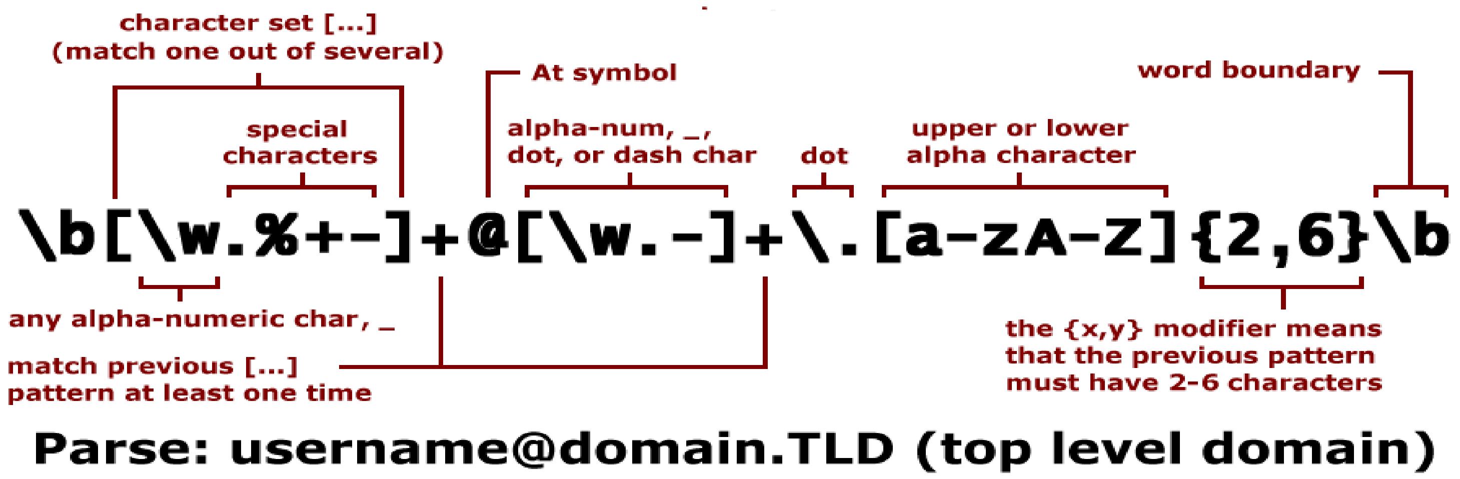
- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

Outline

- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

Regular expressions

- Standard notation for searching strings in text
  - grep, Emacs (Unix)
  - Java, Python, Ruby, Perl, etc.
  - http://regexr.com/



character set [...]
(match one out of several)
At symbol
word boundary
special characters
alpha-num, _, dot, or dash char
dot
upper or lower alpha character
\b[\w.%+-]+@[\w.-]+\.[a-zA-Z]{2,6}\b
any alpha-numeric char, _
match previous [...] pattern at least one time
the {x,y} modifier means that the previous pattern must have 2-6 characters
Parse: username@domain.TLD (top level domain)

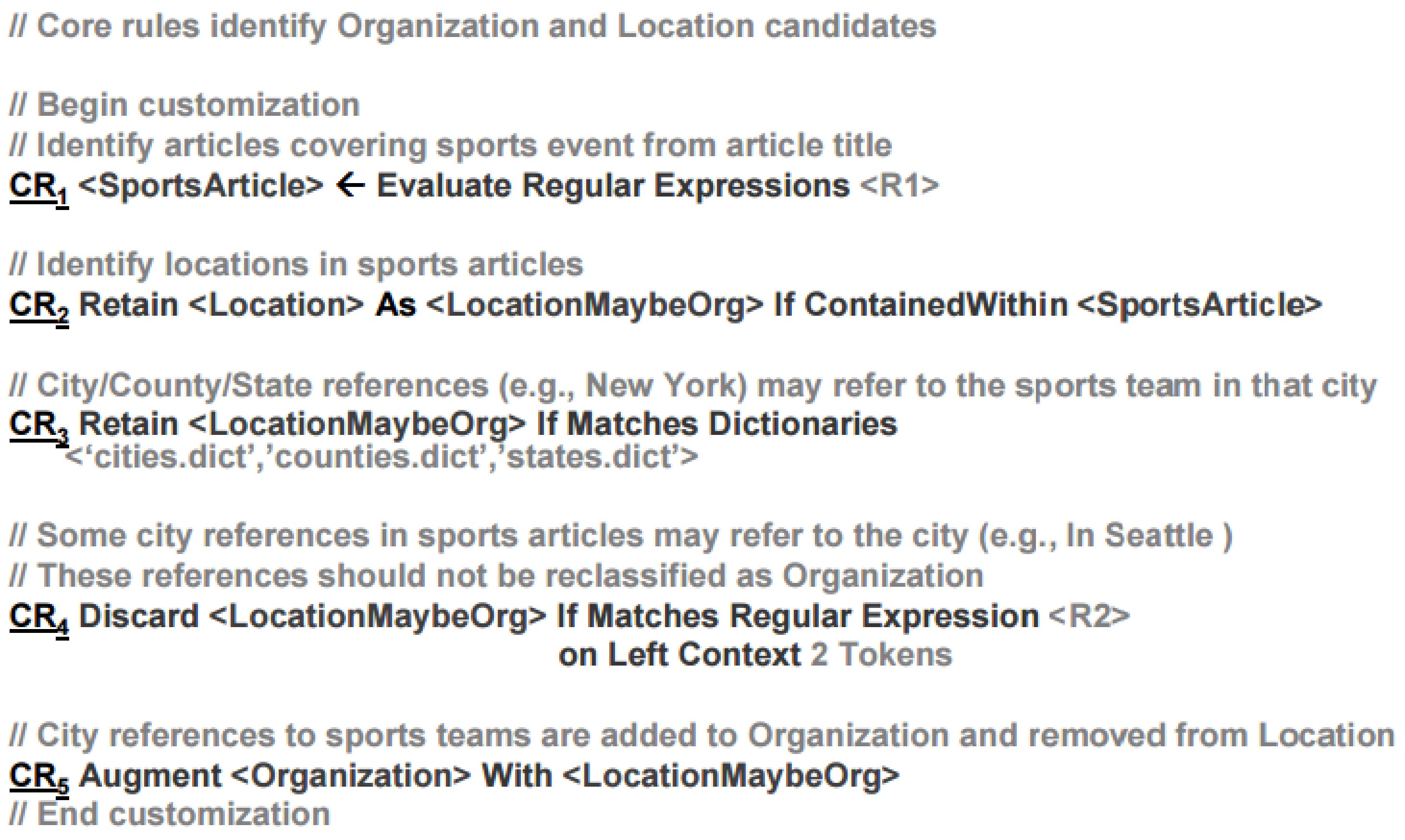
(http://twiki.org/cgi-bin/view/Codev/TWikiPresentation2013x03x07)

Regular expressions

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

(https://xkcd.com/208/)

Motivation – Named-entity recognition



// Core rules identify Organization and Location candidates

// Begin customization
// Identify articles covering sports event from article title
$\underline{CR_1}$ <SportsArticle> ← Evaluate Regular Expressions <R1>

// Identify locations in sports articles
$\underline{CR_2}$ Retain <Location> As <LocationMaybeOrg> If ContainedWithin <SportsArticle>

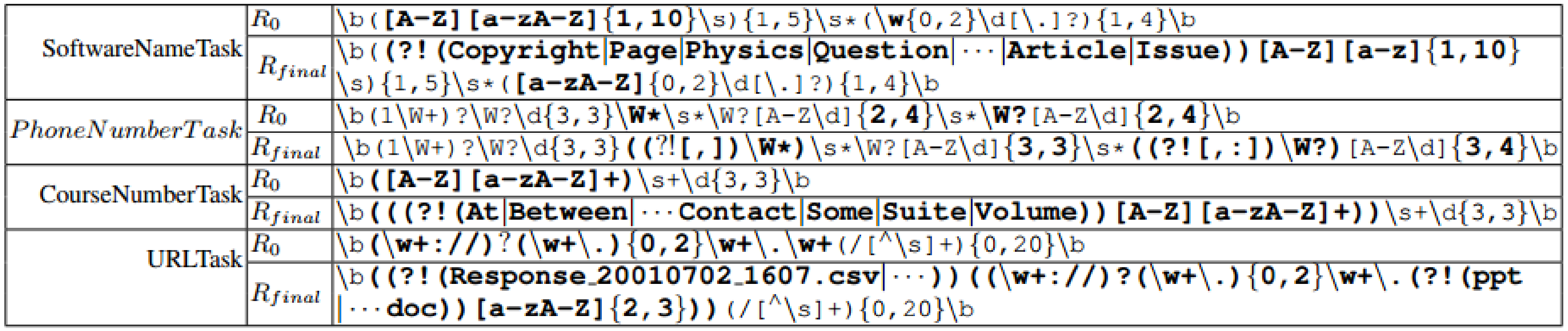
// City/County/State references (e.g., New York) may refer to the sports team in that city
$\underline{CR_3}$ Retain <LocationMaybeOrg> If Matches Dictionaries
<'cities.dict','counties.dict','states.dict'>

// Some city references in sports articles may refer to the city (e.g., In Seattle)
// These references should not be reclassified as Organization
$\underline{CR_4}$ Discard <LocationMaybeOrg> If Matches Regular Expression <R2>
on Left Context 2 Tokens

// City references to sports teams are added to Organization and removed from Location
$\underline{CR_5}$ Augment <Organization> With <LocationMaybeOrg>
// End customization

(http://researcher.watson.ibm.com/researcher/files/us-yunyaoli/emnlp2010-SystemT-namedEntity.pdf)

Motivation – Information extraction



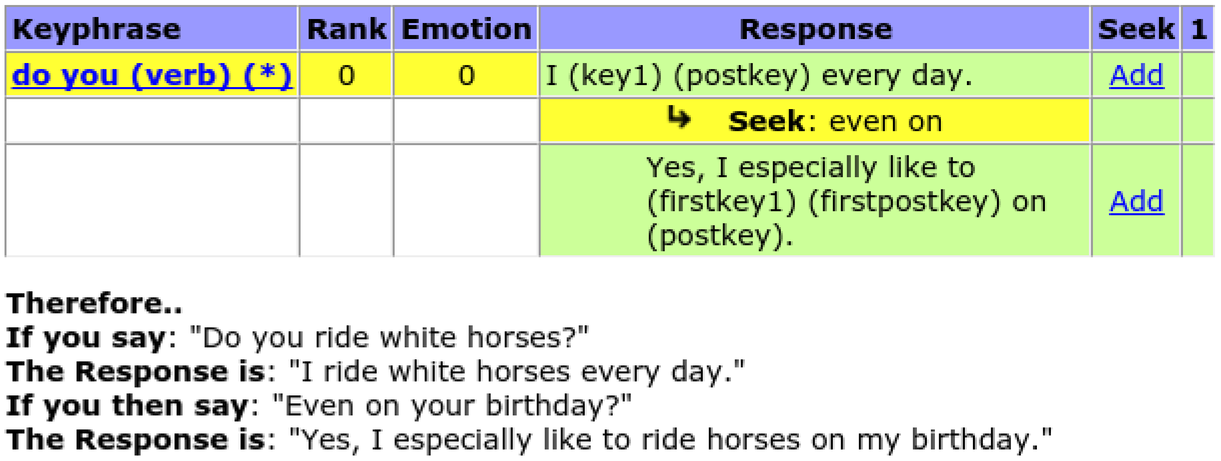
SoftwareNameTask	R_0	\b(**[A-Z][a-zA-Z]{1,10}**\s){1,5}\s*(\**w**{0,2}\d[\\.]?){1,4}\b
	R_{final}	\b(**(?!(Copyright|Page|Physics|Question|···|Article|Issue))[A-Z][a-z]{1,10}**\s){1,5}\s*(**[a-zA-Z]**{0,2}\d[\\.]?){1,4}\b
PhoneNumberTask	R_0	\b(1\W+)?\W?\d{3,3}**\W***\s*\W?[A-Z\d]**{2,4}**\s***\W?**[A-Z\d]**{2,4}**\b
	R_{final}	\b(1\W+)?\W?\d{3,3}**((?![,])\W*)**\s*\W?[A-Z\d]**{3,3}**\s***((?![,:])\W?)**[A-Z\d]**{3,4}**\b
CourseNumberTask	R_0	\b**([A-Z][a-zA-Z]+)**\s+\d{3,3}\b
	R_{final}	\b**(((?!(At|Between|···Contact|Some|Suite|Volume))[A-Z][a-zA-Z]+))**\s+\d{3,3}\b
URLTask	R_0	\b**(\w+://)**?**(\w+\\.)**{**0,2**}**\w+**\\.**\w+**(/[^\s]+){0,20}\b
	R_{final}	\b**((?!(Response_20010702_1607.csv|···))((\w+://)?(\w+\\.){0,2}\w+\\.(?!(ppt|···doc))[a-zA-Z]{2,3}))**(/[^\s]+){0,20}\b

Table 4: Sample Regular Expressions Learned by ReLIE(R_0: input regex; R_{final}: final regex learned; the parts of R_0 modified by ReLIE and the corresponding parts in R_{final} are highlighted.)

(http://www.aclweb.org/anthology/D08-1003)

Motivation - Chatterbot

- ELIZA (Weizenbaum 1966)



Keyphrase	Rank	Emotion	Response	Seek	1
do you (verb) (*)	0	0	I (key1) (postkey) every day.	Add	
			↳ **Seek**: even on		
			Yes, I especially like to (firstkey1) (firstpostkey) on (postkey).	Add	

Therefore..
If you say: "Do you ride white horses?"
The Response is: "I ride white horses every day."
If you then say: "Even on your birthday?"
The Response is: "Yes, I especially like to ride horses on my birthday."

(http://www.personalityforge.com/book-expert.php)
(http://www.codeproject.com/Articles/18109/Building-an-AI-Chatbot-using-a-Regular-Expression)
(http://www.majidkhosravi.com/chat-bot/)

Motivation – Spell checking

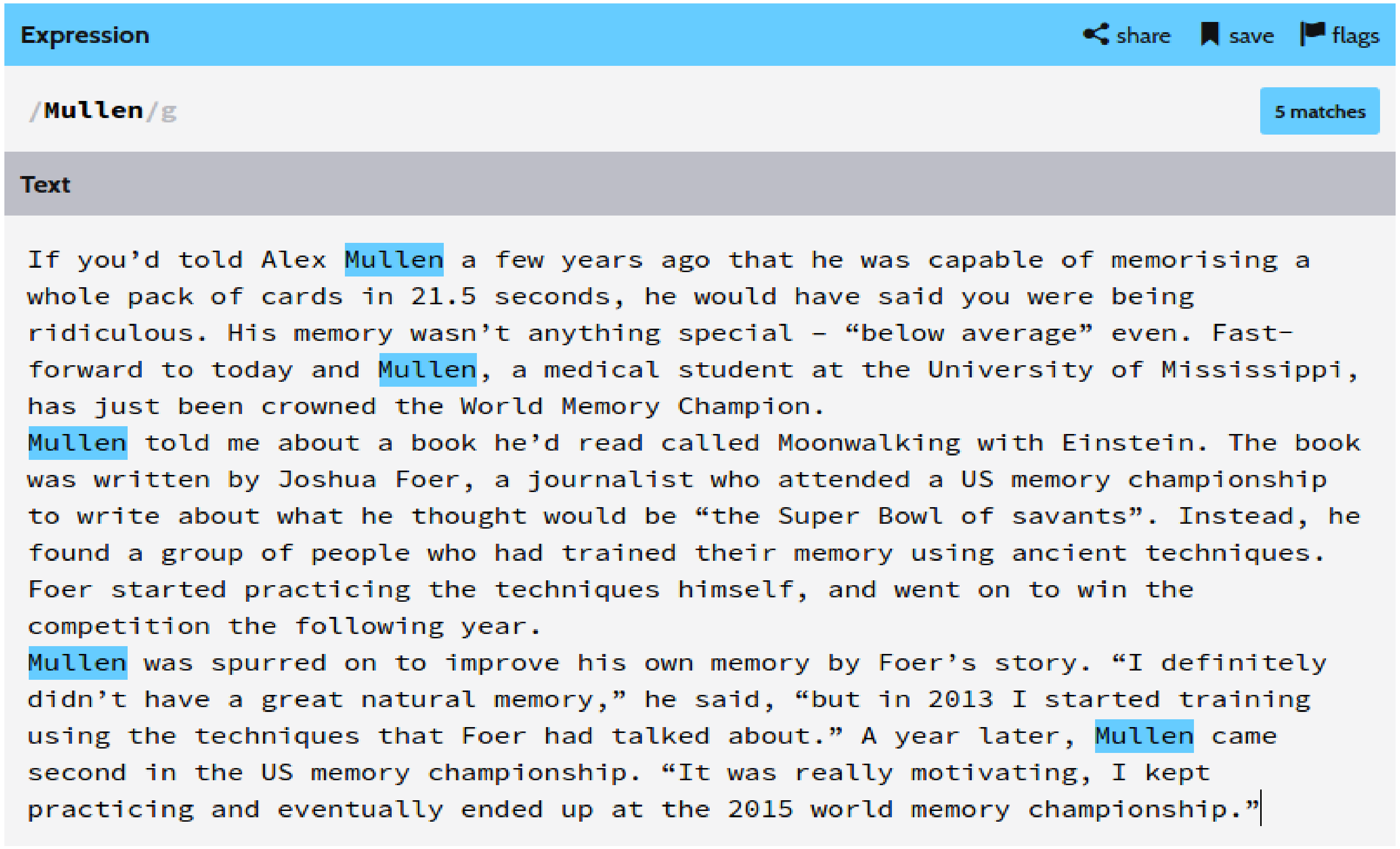
- Checking spelling and grammar
- Suggesting alternatives for the errors

Regular expressions

- Specify classes of strings
- Consider a string as a sequence of alphanumeric characters

- **Patterns** to be searched for
- **Documents** of text to search through

Searching particular words


Expression

share save flags

/Mullen/g

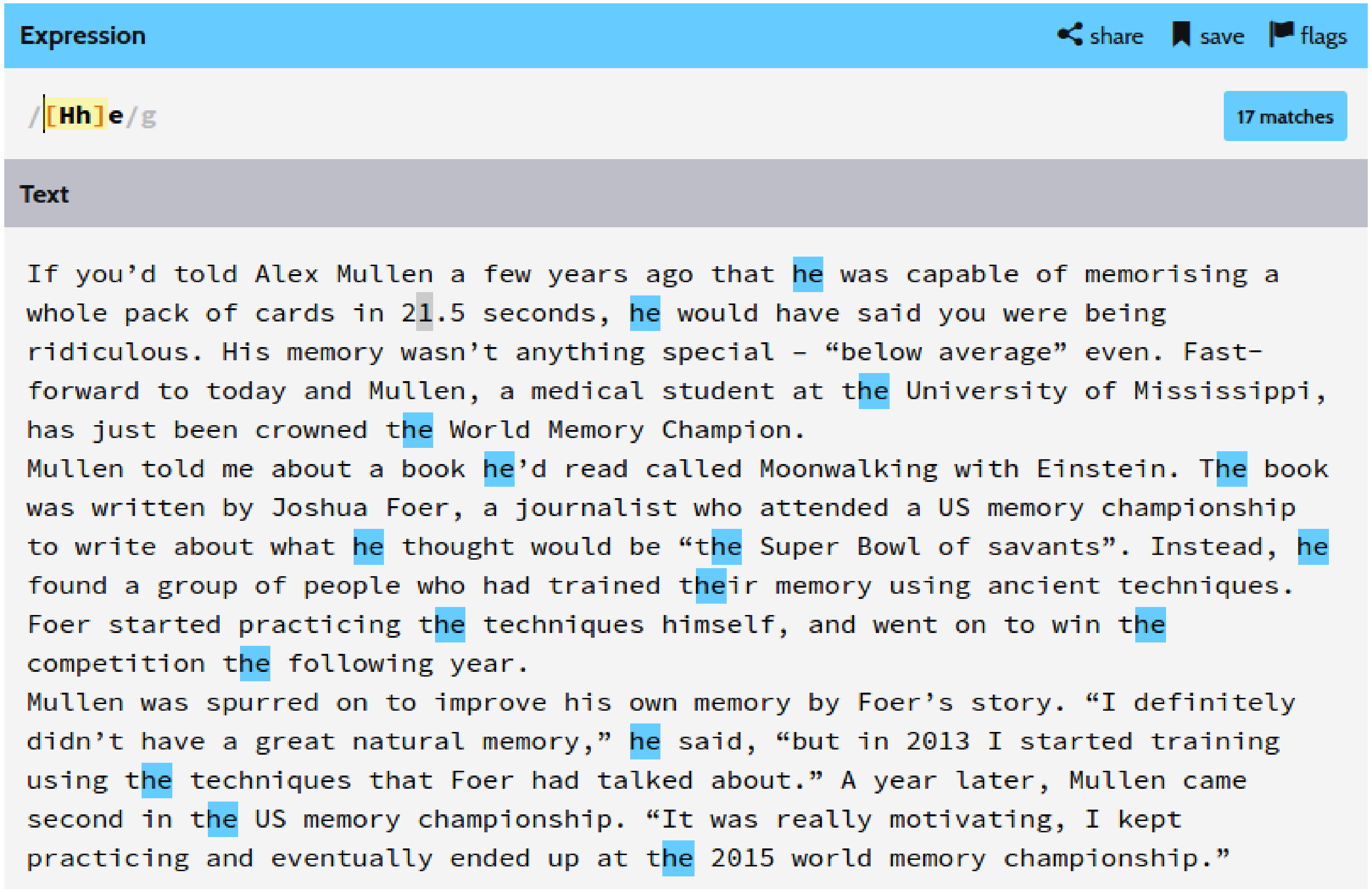
5 matches

Text

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

(http://regexr.com/)

Searching particular words

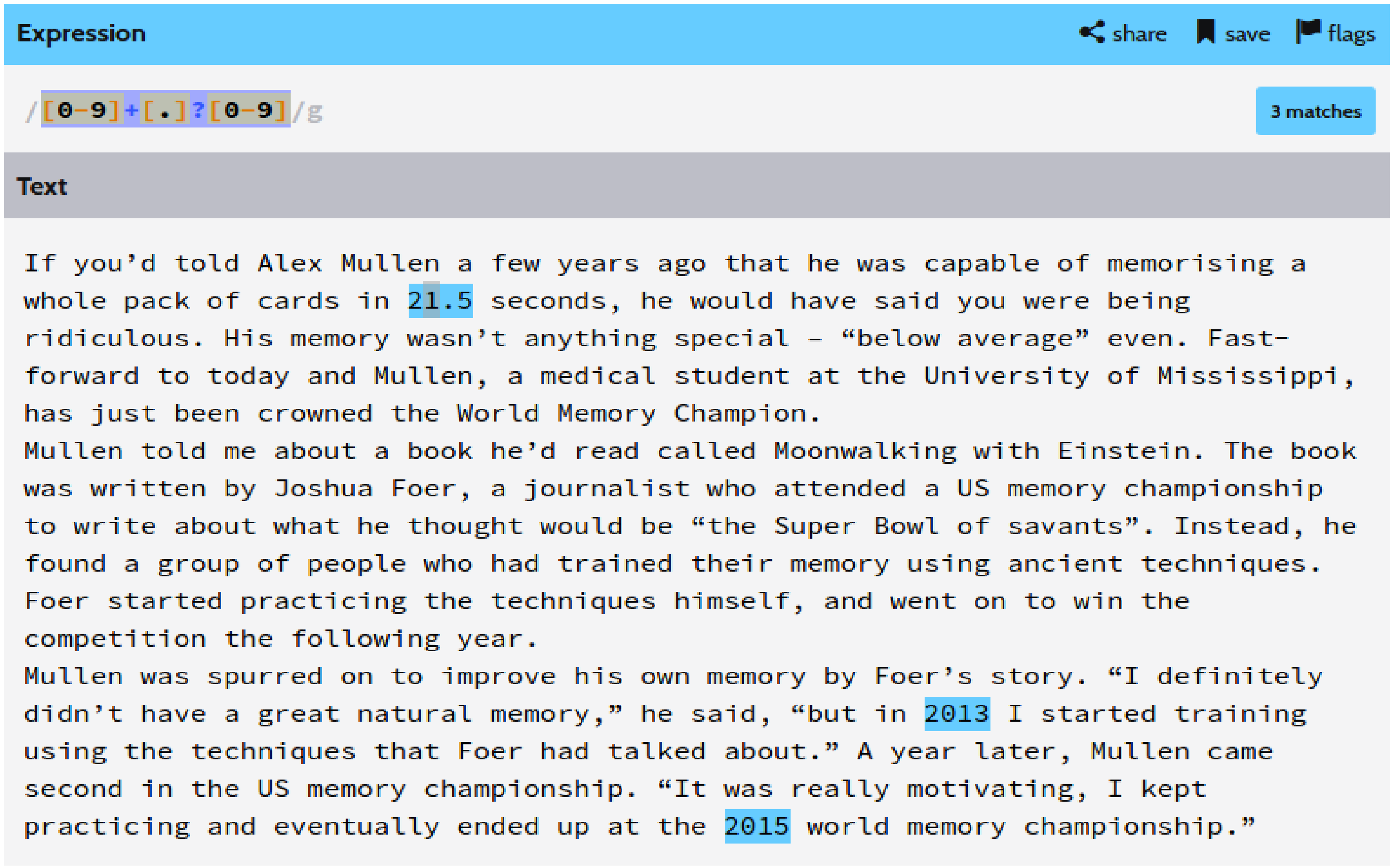


```
/[Hh]e/g
```

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

(http://regexr.com/)

Searching numbers

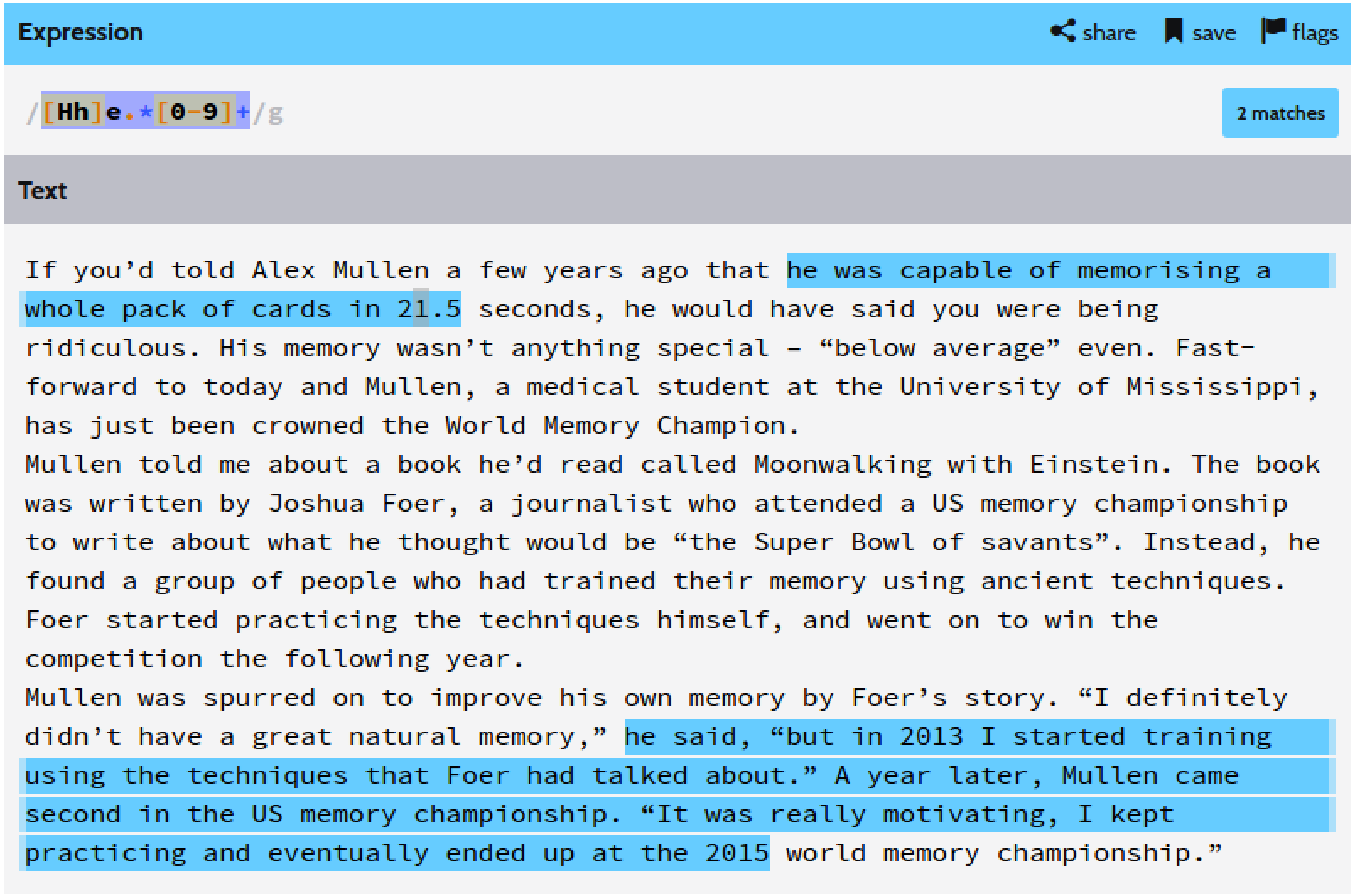


```
/[0-9]+[.]?[0-9]/g
```

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

(http://regexr.com/)

Searching passages with numbers

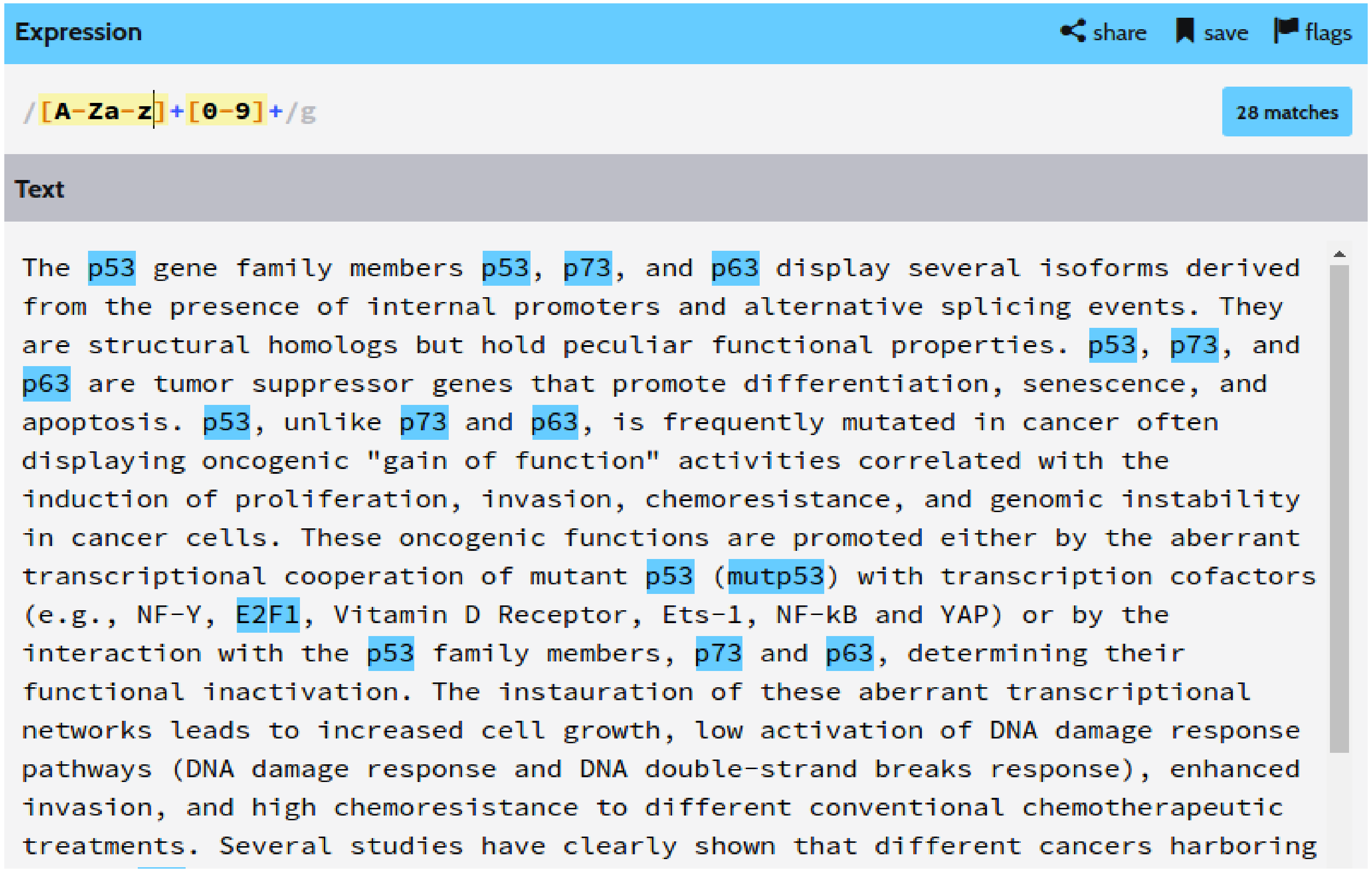


```
/[Hh]e.*[0-9]+/g
```

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

(http://regexr.com/)

Named-entity recognition


Expression share save flags

```
/[A-Za-z]+[0-9]+/g
```

28 matches

Text

The p53 gene family members p53, p73, and p63 display several isoforms derived from the presence of internal promoters and alternative splicing events. They are structural homologs but hold peculiar functional properties. p53, p73, and p63 are tumor suppressor genes that promote differentiation, senescence, and apoptosis. p53, unlike p73 and p63, is frequently mutated in cancer often displaying oncogenic "gain of function" activities correlated with the induction of proliferation, invasion, chemoresistance, and genomic instability in cancer cells. These oncogenic functions are promoted either by the aberrant transcriptional cooperation of mutant p53 (mutp53) with transcription cofactors (e.g., NF-Y, E2F1, Vitamin D Receptor, Ets-1, NF-kB and YAP) or by the interaction with the p53 family members, p73 and p63, determining their functional inactivation. The instauration of these aberrant transcriptional networks leads to increased cell growth, low activation of DNA damage response pathways (DNA damage response and DNA double-strand breaks response), enhanced invasion, and high chemoresistance to different conventional chemotherapeutic treatments. Several studies have clearly shown that different cancers harboring

(http://regexr.com/)

Outline

- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

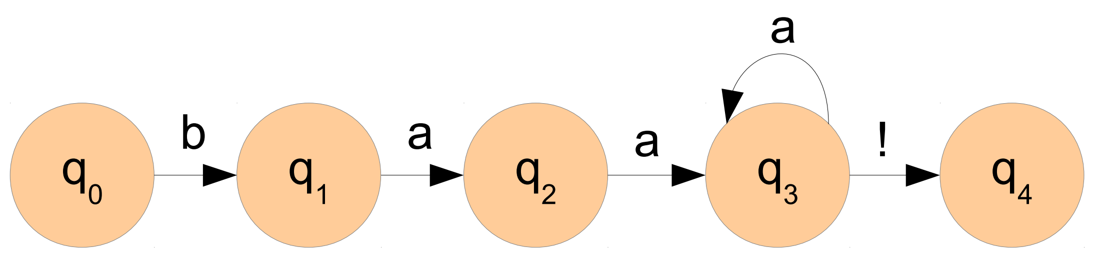
Finite-State Automata (FSA)

- Regular expression
  - is one way to describe FSA
  - is one way to characterize a regular language

- Regular language can be characterized by
  - Regular expressions
  - FSAs
  - Regular grammars

Finite-State Automata (FSA)

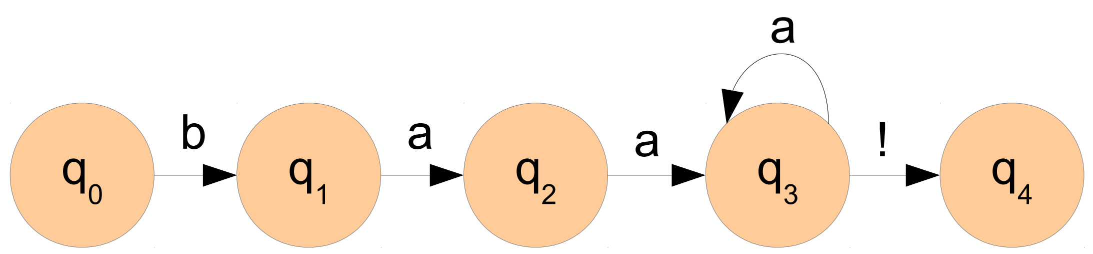
- FSAs are composed of
  - Vertices (nodes)
  - Arcs (links)



string?

Finite-State Automata (FSA)

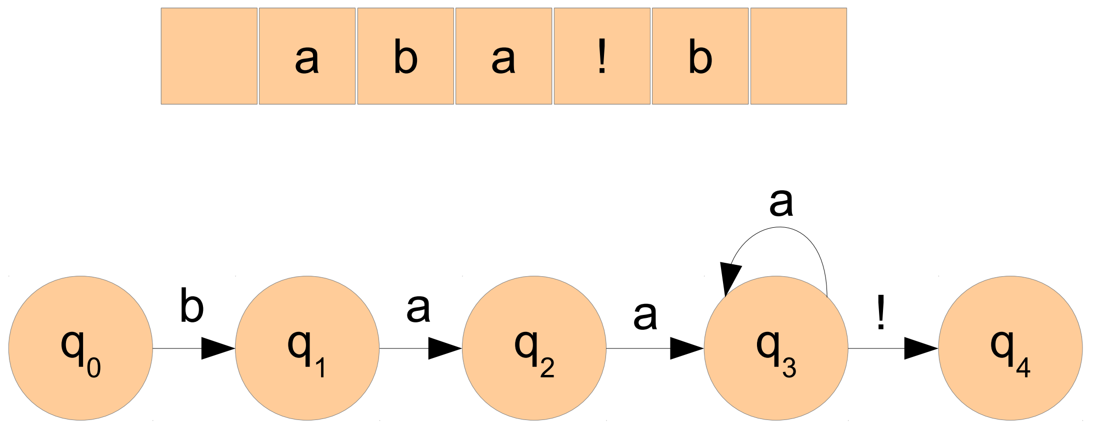
- FSAs are composed of
  - Vertices (nodes)
  - Arcs (links)



baa!
baaa!
baaaa!
…

/baa+!/

FSA – accepting/rejecting strings



state transition table

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

FSA – accepting/rejecting strings

- Strings accepted
  - baa!
  - baaa!

- Strings rejected
  - abc
  - aba!
  - ba!
  - baaa

state transition table

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

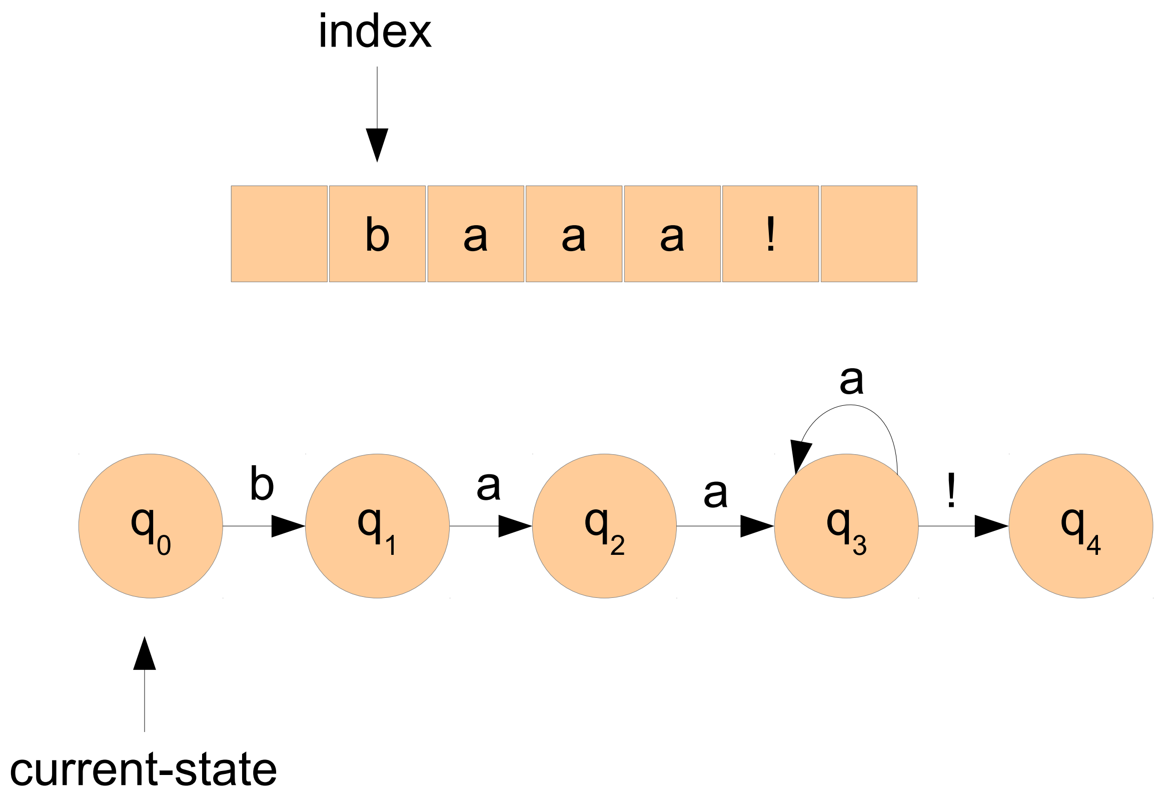
Formalization of FSA

- $Q = q_0 q_1 q_2 \dots q_{N-1}$: finite set of N states
- Σ : finite input alphabet of symbols
- q_0 : the start state
- F: the set of final states, $F \subseteq Q$
- $\delta(q, i)$: the transition function ($q \in Q$; input symbol $i \in \Sigma$)

Example: formalization for the „sheeptalk"

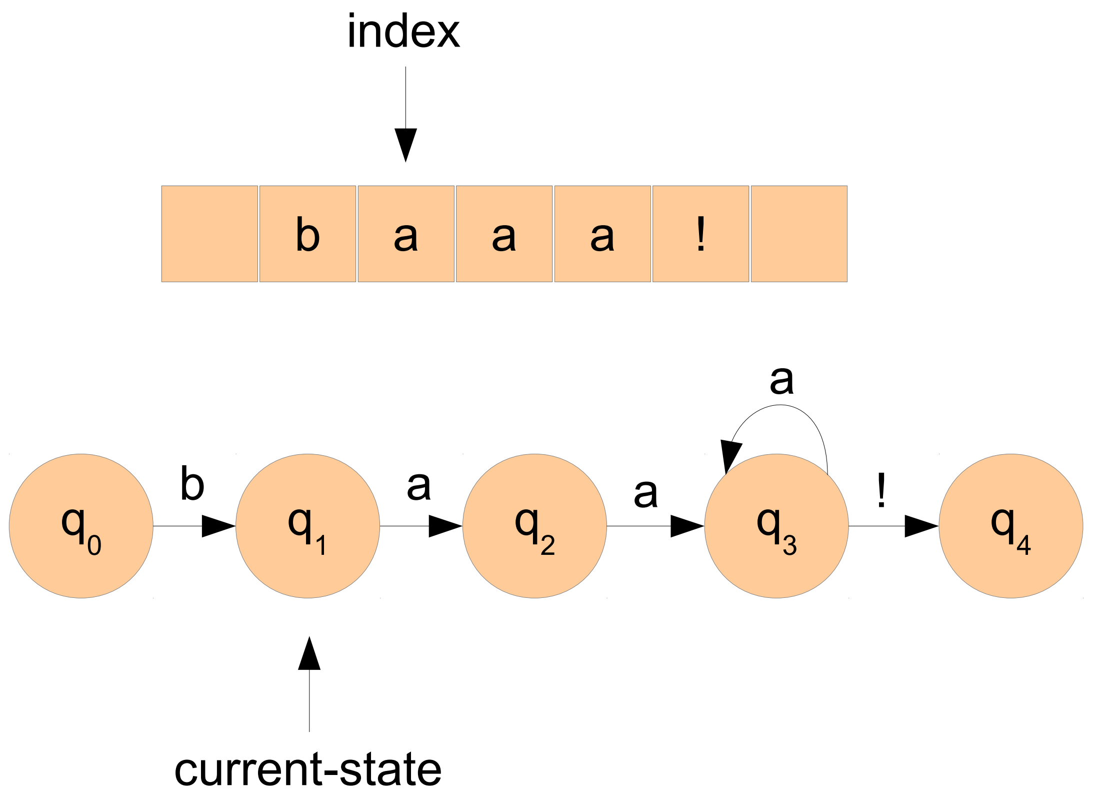
- $Q = q_0 q_1 q_2 q_3 q_4$
- $\Sigma = \{a, b, !\}$
- q_0
- $F = q_4$
- $\delta(q, i)$ defined by the transition table

D-RECOGNIZE algorithm (step 1)



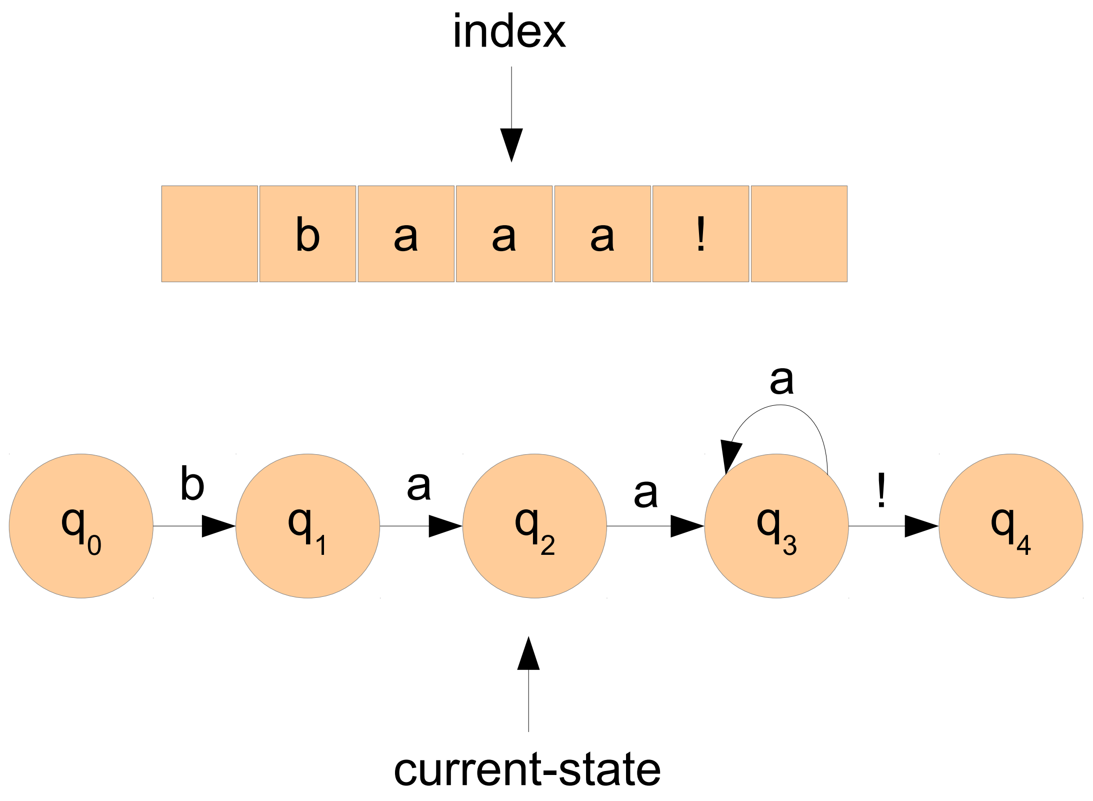
	Input		
State	b	a	!
0	1	ø	ø
1	ø	2	ø
2	ø	3	ø
3	ø	3	4
4	ø	ø	ø

D-RECOGNIZE algorithm (step 2)



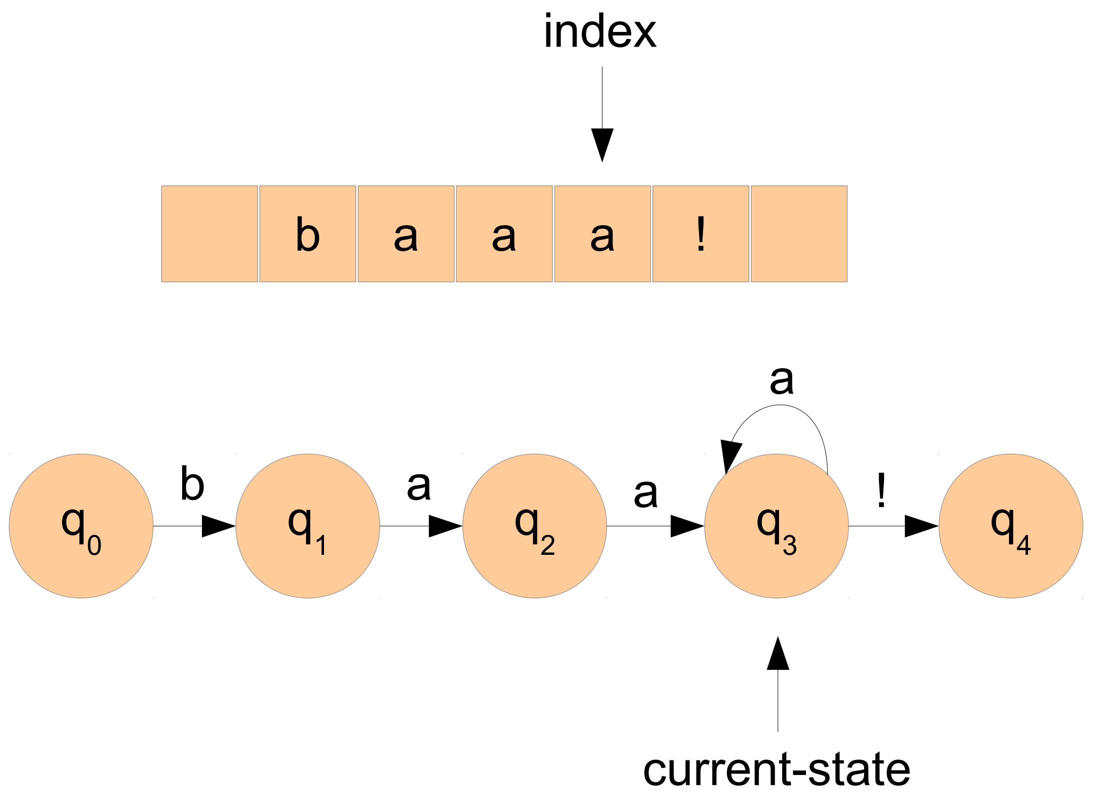
	Input		
State	b	a	!
0	1	ø	ø
1	ø	2	ø
2	ø	3	ø
3	ø	3	4
4	ø	ø	ø

D-RECOGNIZE algorithm (step 3)



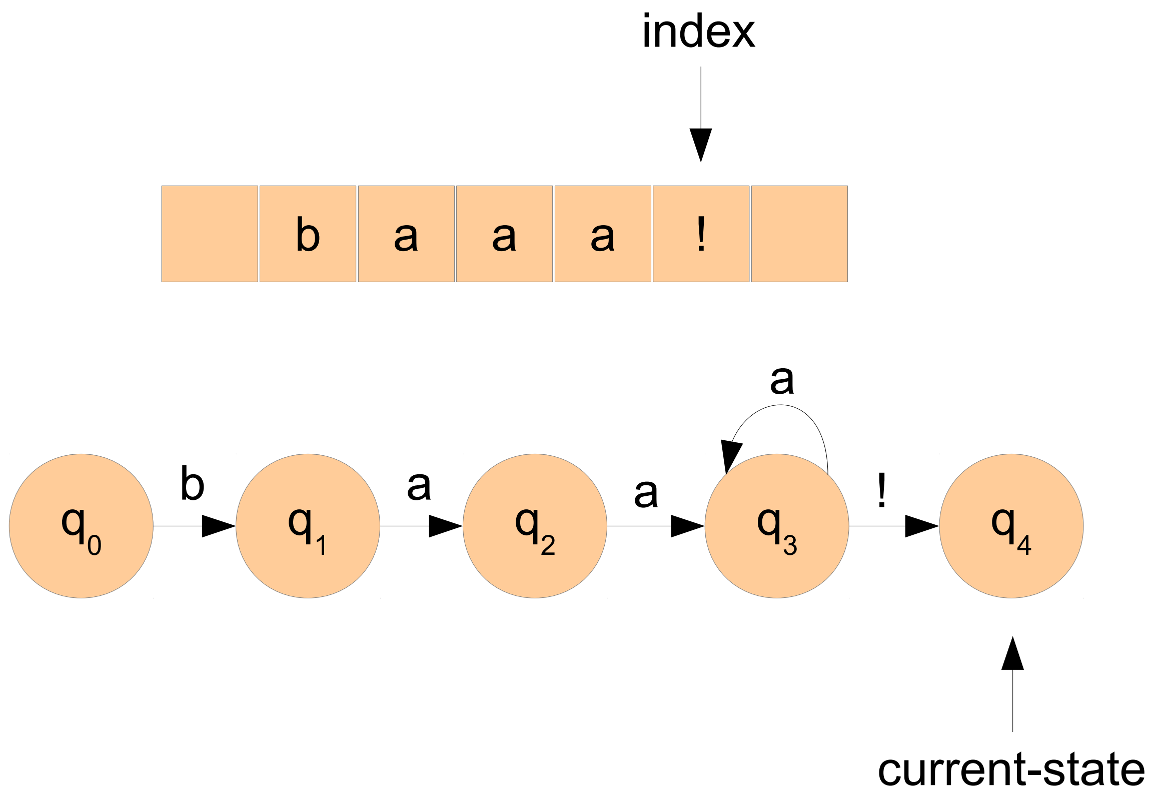
	Input		
State	b	a	!
0	1	ø	ø
1	ø	2	ø
2	ø	3	ø
3	ø	3	4
4	ø	ø	ø

D-RECOGNIZE algorithm (step 4)



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

D-RECOGNIZE algorithm (step 5)



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

- q_4 is a final state and input is over, then the string is accepted!
(Pseudo-code for the algorithm in Jurafski and Martin book.)

Formal language

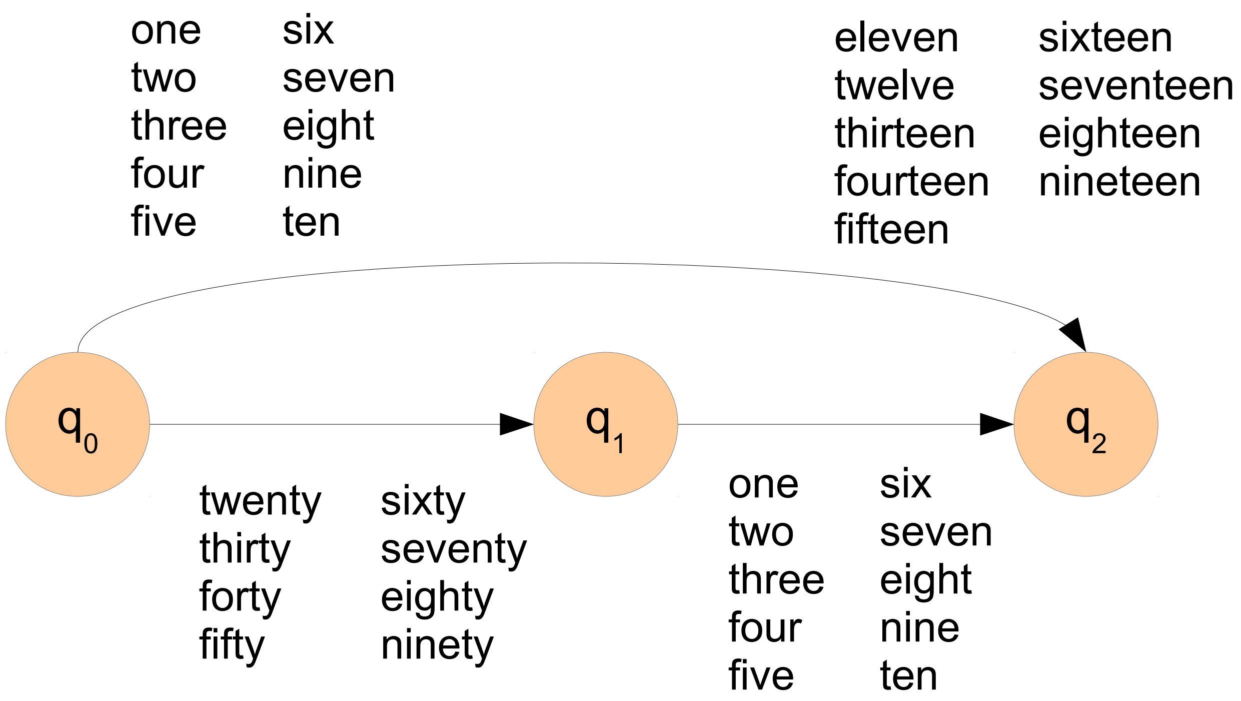
- A model that can both generate and recognize all and only the strings given by its definition.
- An automaton can describe an infinite set with a closed form.

- „sheeptalk“ model „m“:
  - $\Sigma = \{b,a,!\}$
  - L(m) = formal language characterized by „m“
  - L(m) = {baa!,baaa!,baaaa!,....}

Formal language ≠ natural language

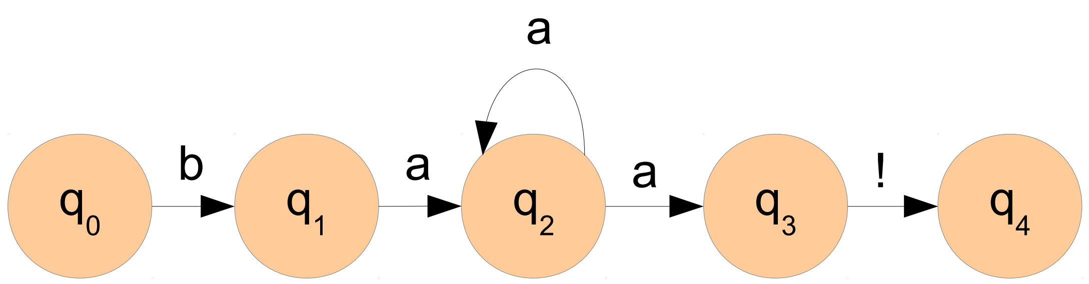
- But we can use formal languages to model parts of a natural language
  - Phonology, morphology or syntax

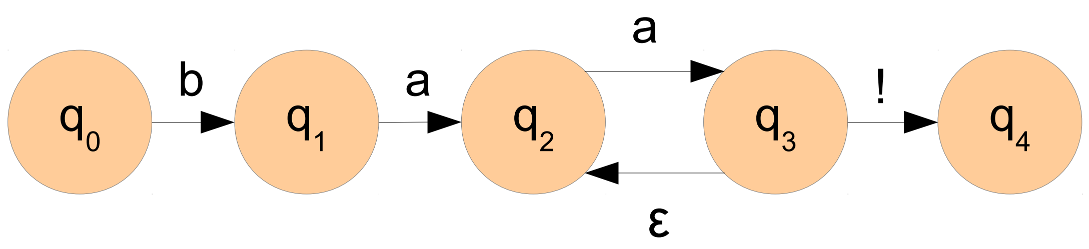
Modelling amounts of money (1-99 cents)



(Check example for dollars and cents in Jurafski & Martin book.)

Non-deterministic FSAs





	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

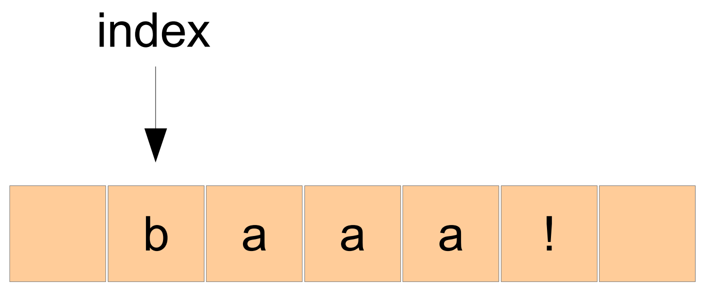
Solutions for non-deterministic FSAs

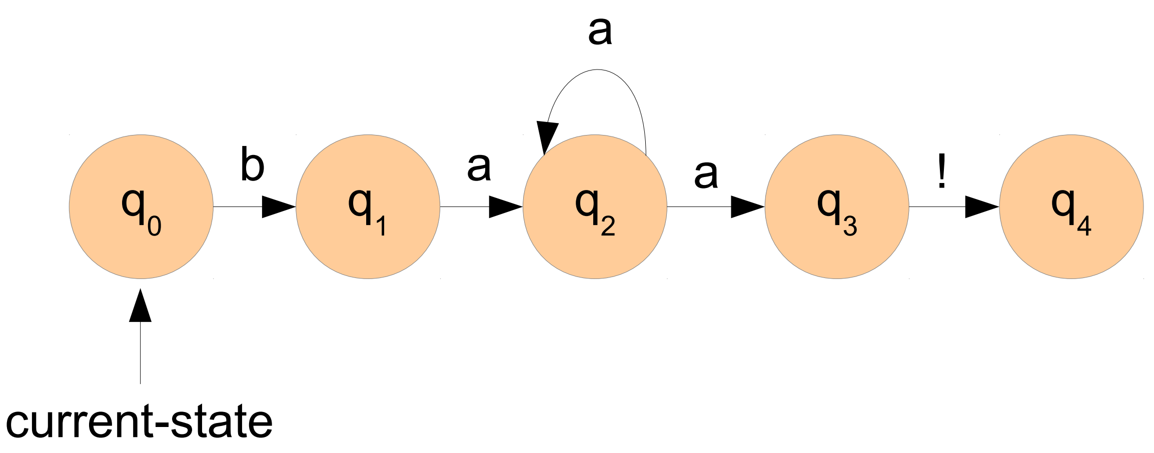
- Backup
  - Keep track of points with many choices
- Look-ahead
  - Look ahead the input to decide the path
- Parallelism
  - Explore many paths in parallel

ND-RECOGNIZE algorithm (step 1)

Agenda

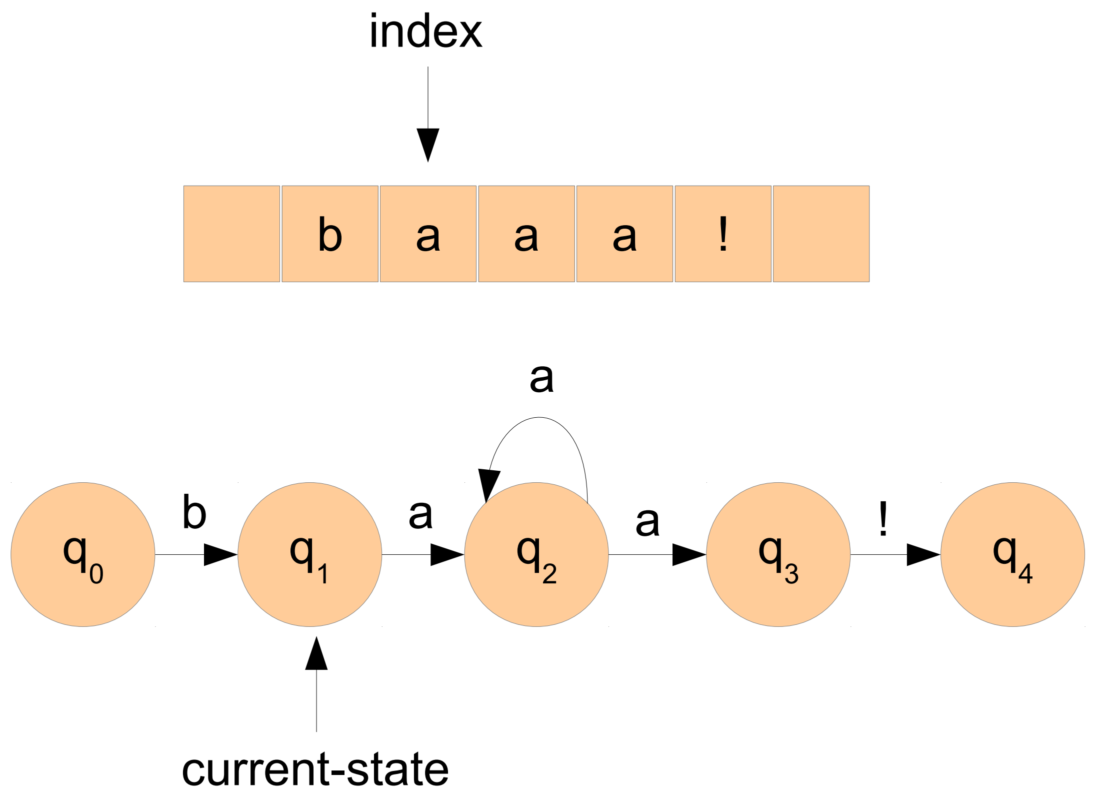
q_0,b





	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

ND-RECOGNIZE algorithm (step 2)



Agenda

~~q_0,b~~

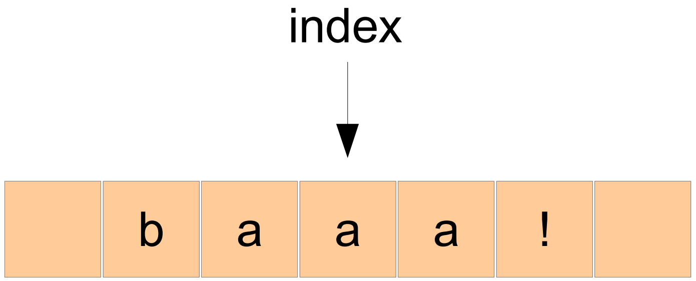
q_1,a

<table><tr><td></td><td colspan="4">Input</td></tr></table>

	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

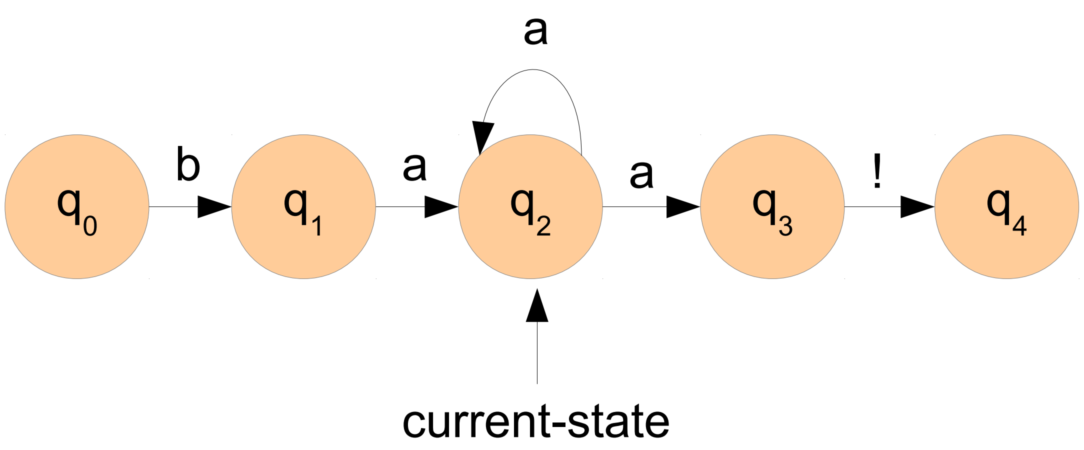
Hasso Plattner Institut

ND-RECOGNIZE algorithm (step 3)



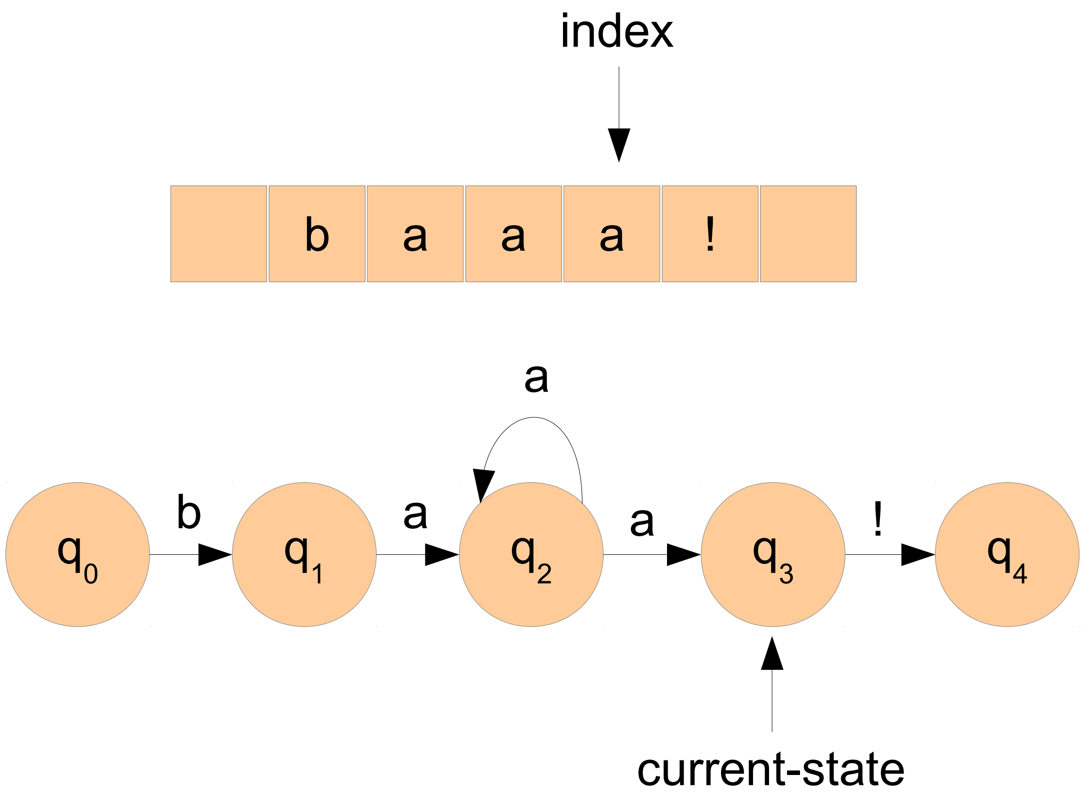
Agenda

~~q_0,b~~
~~q_1,a~~
q_2,a



	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

ND-RECOGNIZE algorithm (step 4)

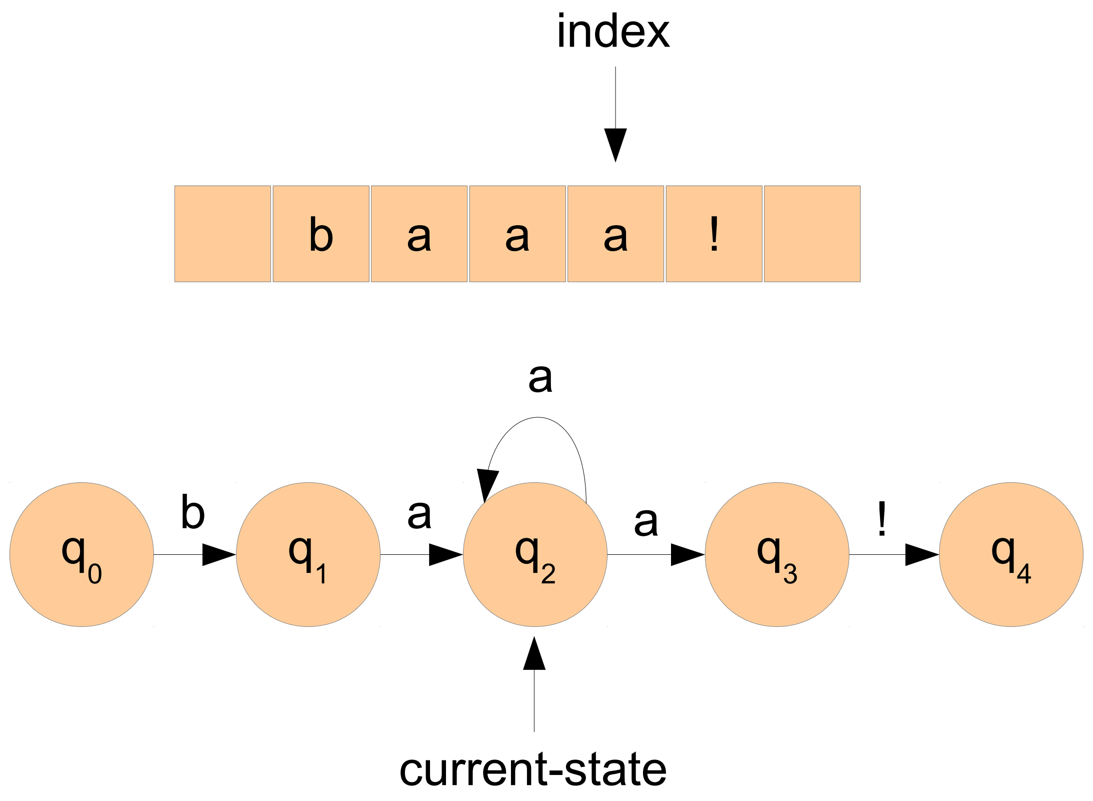


Agenda

~~q_0,b~~
~~q_1,a~~
~~q_2,a~~
q_3,a
q_2,a

	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

ND-RECOGNIZE algorithm (step 5)

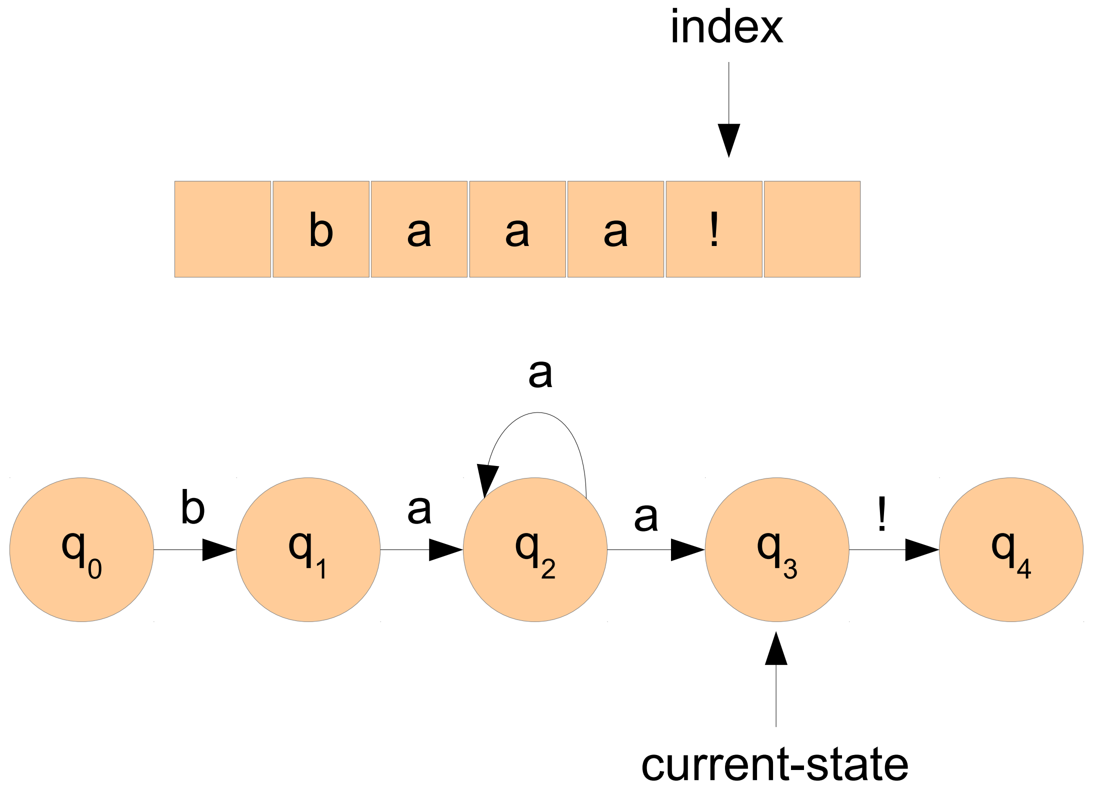


Agenda

~~q_0,b~~
~~q_1,a~~
~~q_2,a~~
~~q_3,a~~
q_2,a

	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

ND-RECOGNIZE algorithm (step 6)

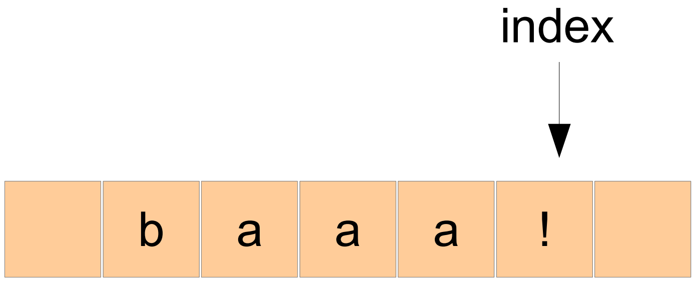


Agenda

~~q_0,b~~	q_3,!
~~q_1,a~~	q_2,!
~~q_2,a~~	
~~q_3,a~~	
~~q_2,a~~	

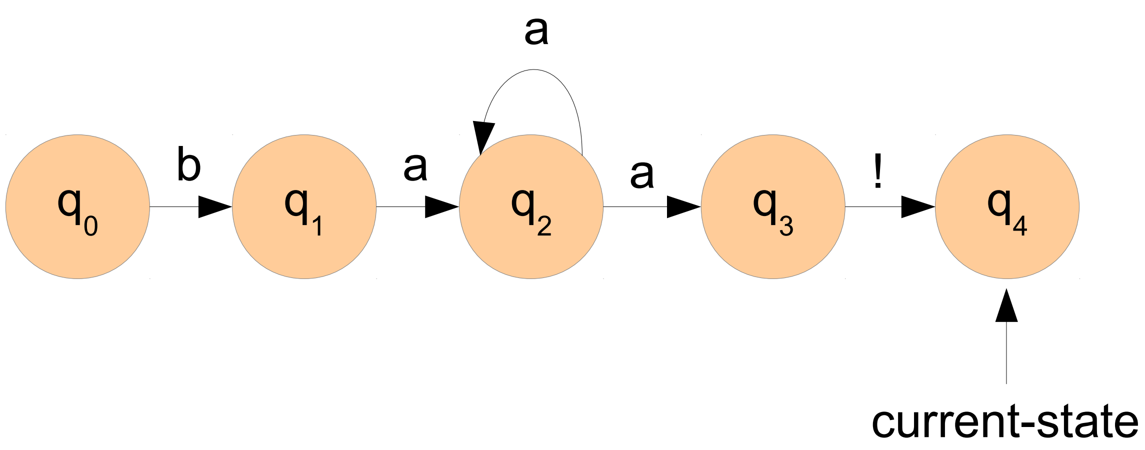
	Input			
State	b	a	!	ε
0	1	ø	ø	ø
1	ø	2	ø	ø
2	ø	2,3	ø	ø
3	ø	ø	4	ø
4	ø	ø	ø	ø

ND-RECOGNIZE algorithm (step 7)



Agenda

~~q_0,b~~
~~q_1,a~~
~~q_2,a~~
~~q_3,a~~
~~q_2,a~~
~~q_3,!~~
q_2,!



	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

- q_4 is a final state and input is over, then the string is accepted!

(Pseudo-code for the algorithm in Jurafski and Martin book.)

State-Space Search

- The algorithm creates a space of possible solutions
  - Path (solutions) are examined and accepted or rejected
- The order of the exploration plays an important role
  - But there is no way of knowing it beforehand
- The ordering is not specified in the ND-RECOGNIZE algorithm
  - Stack: depth-first-search, last in first out (LIFO)
  - Queue: breadth-first search, first in first out (FIFO)
  - For complex/large problems, dynamic programming is used

DFSA vs. NFSA

- There is a DFSA for every NFSA
  - But it needs many more nodes
  - As much as 2^N, where „N“ is the number of distinct sets in the original NFSA

Regular language - definition

- Σ : alphabet, set all symbols in the language
- Σ^* : infinite set of all possible strings formed from Σ
- Ø : empty set, which is a regular language

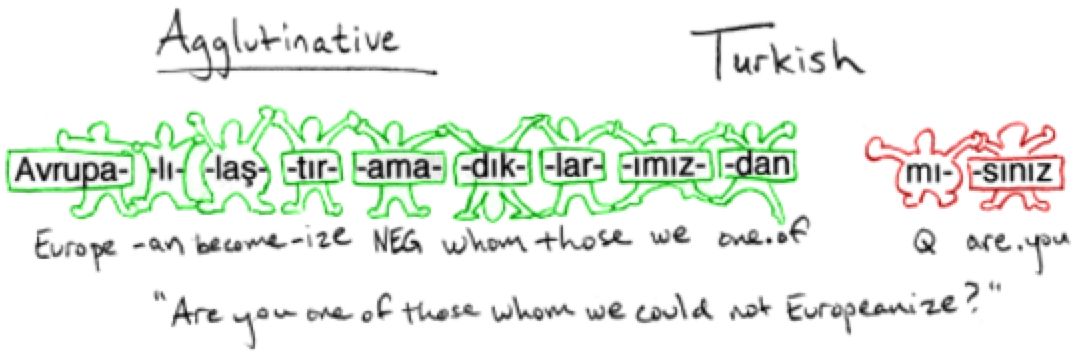
- L_1 and L_2 are languages, then:
  - Concatenation: $L_1 \cdot L_2$
  - Union or disjunctions: $L_1 \cup L_2$
  - Kleene closure: $L_1{}^*$

Outline

- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

Morphology

- The study of internal structures of words and how they can be modified
- Parsing complex words into their components

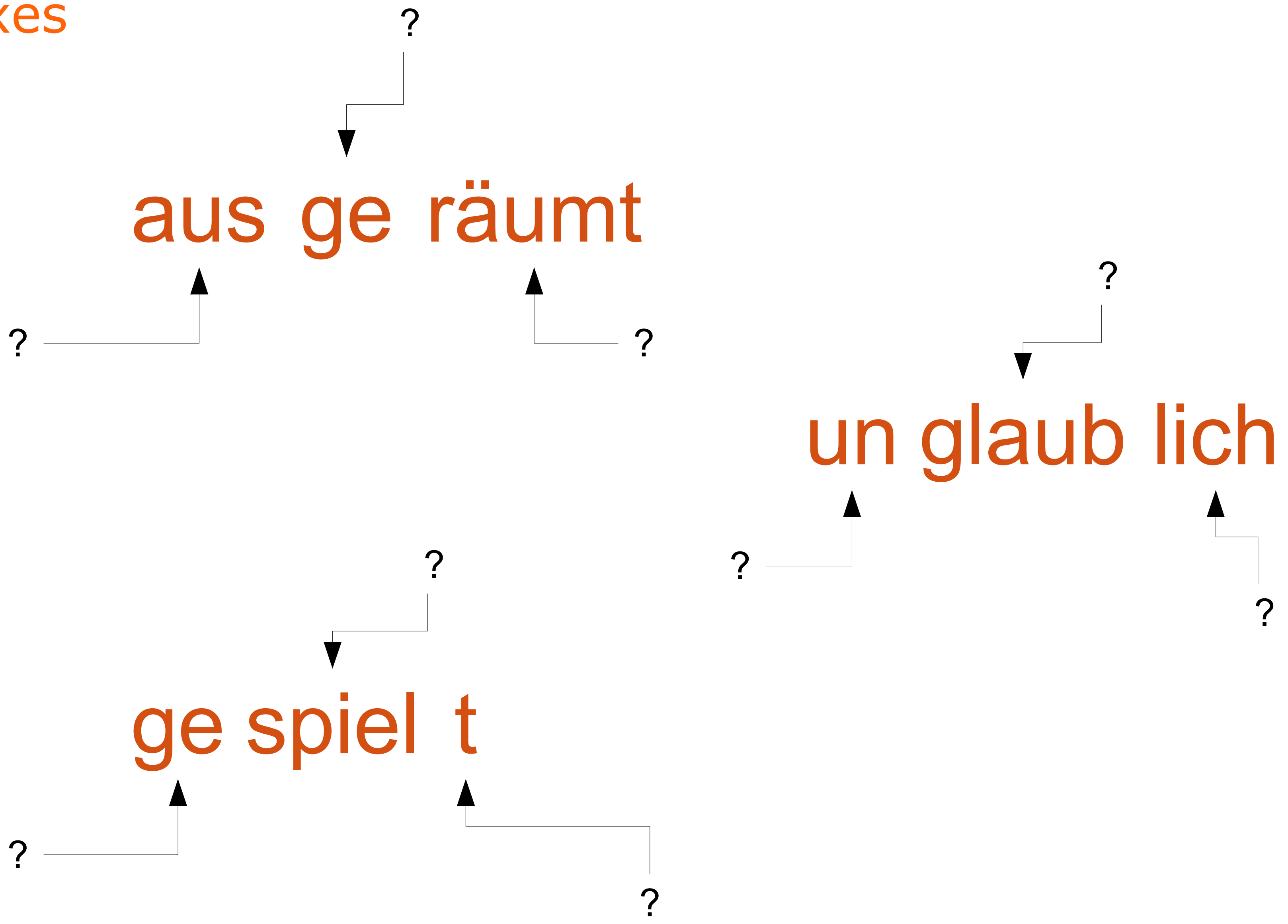

Agglutinative
Turkish
Avrupa- -lı- -laş- -tır- -ama- -dık- -lar- -ımız- -dan
mı- -sınız
Europe -an become -ize NEG whom those we one.of
Q are.you
"Are you one of those whom we could not Europeanize?"

(http://allthingslinguistic.com/post/50939757945/morphological-typology-illustrations-from)

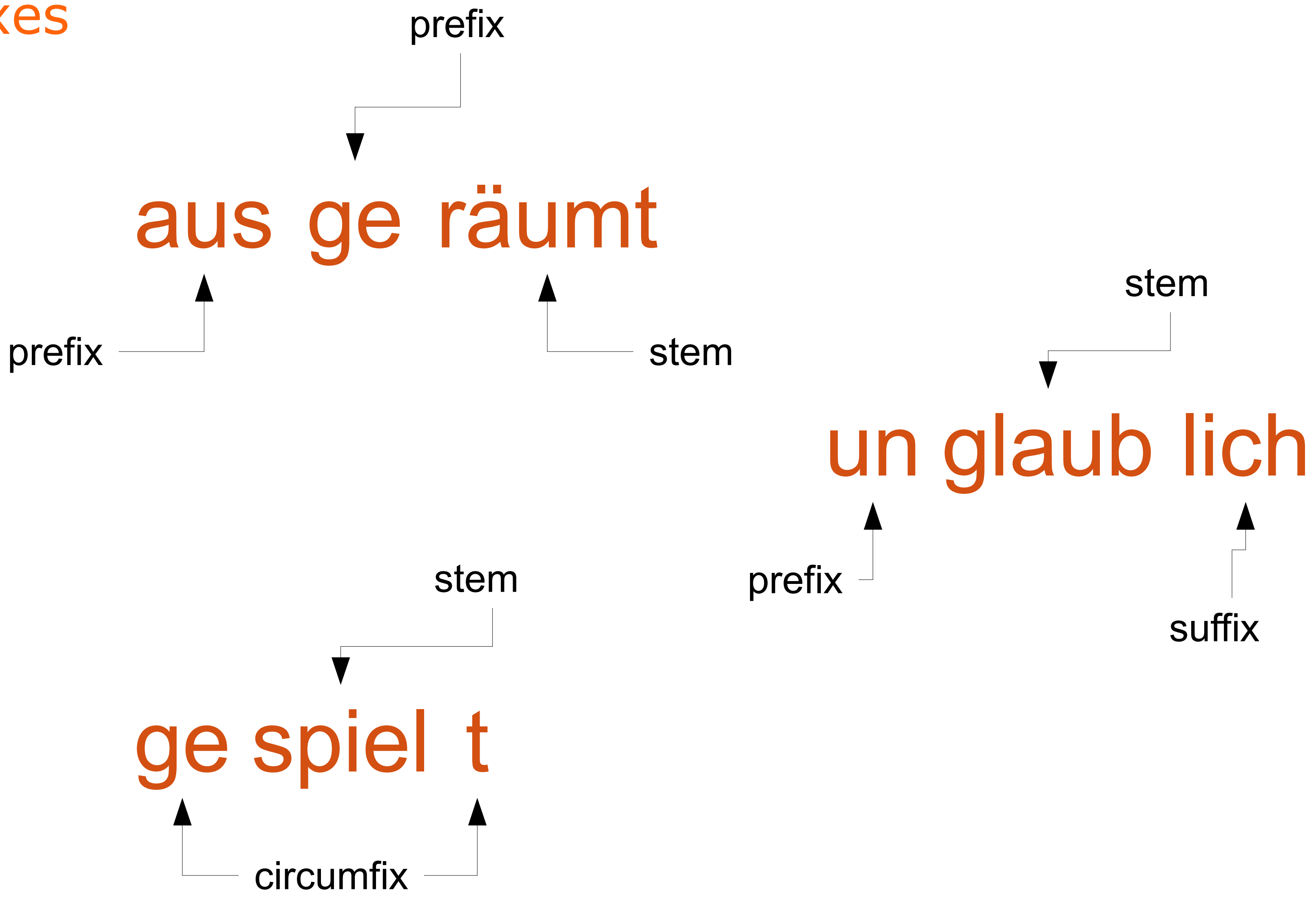
Morphology

- Study of the morphemes
  - A morpheme is the minimal meaning-bearing unit in a language
  - Stem: main morpheme
  - Affixes: additional meanings
    - Prefix (before)
    - Suffix (after)
    - Circumfix (both)
    - Infix (middle)

Affixes



Affixes


prefix
aus ge räumt
prefix
stem
stem
un glaub lich
prefix
suffix
stem
ge spiel t
circumfix

Combining morphemes

- Inflection
  - Resulting word is of the same class of the original word
  - Plurals, conjugation of verbs
  - *cat (cats), play(played)*
- Derivation
- Compounding
- Cliticization

Combining morphemes

- Inflection
- Derivation
  - Resulting word is of a distinct class of the original word
  - For instance, verbs to nouns or to adjectives
  - *compute (computation, computational)*
- Compounding
- Cliticization

Combining morphemes

- Inflection
- Derivation
- Compounding
  - Combination of multiple word stems
  - *doghouse*
- Cliticization

Combining morphemes

- Inflection
- Derivation
- Compounding
- Cliticization
  - Combination of a word stem with a clitic
  - Clitic: a morpheme that acts like a word but is reduced and attached to another word
  - *I've, l'opera*

Outline

- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

Morphological parsing

- Breaking down words into components and building a structured representation
  - English:
    - cats → cat +N +Pl
    - caught → catch +V +Past
  - Spanish:
    - vino (came) → venir +V +Perf +3P +Sg
    - vino (wine) → vino +N +Masc +Sg

Motivation for morphological parsing

- Information retrieval
  - Normalize verb tenses, plurals, grammar cases

- Machine translation
  - Translation based on the stem

Stemming vs. Lemmatization

- Stemming: stripping off word endings (rule-based)
  - foxes → fox
  - going → go

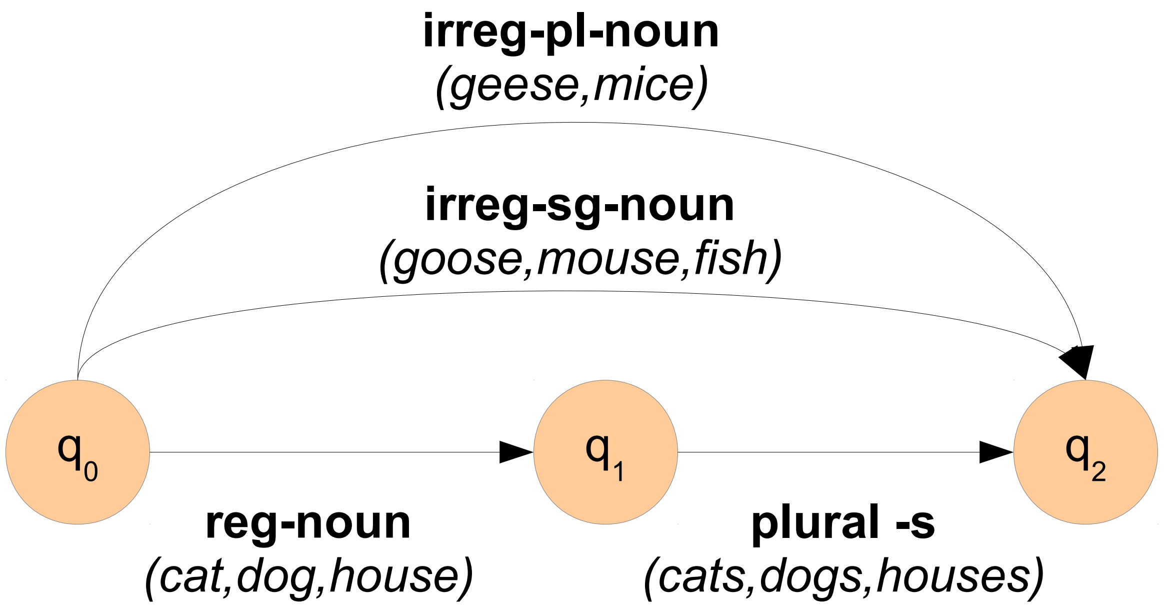
- Lemmatization: mapping the word to its lemma (lexicon-based)
  - sang, sung → sing
  - going, went, goes → go

Morphological parsing

- Resources
  - Lexicon
    - List of all stems and affixes
  - Morphotactics
    - A model of morpheme ordering in a word
    - e.g., plurals are suffixes in English
  - Ortographic rules
    - Rules for changing in the words when combining morphemes
    - e.g., city → cities

Finite-State Lexicon

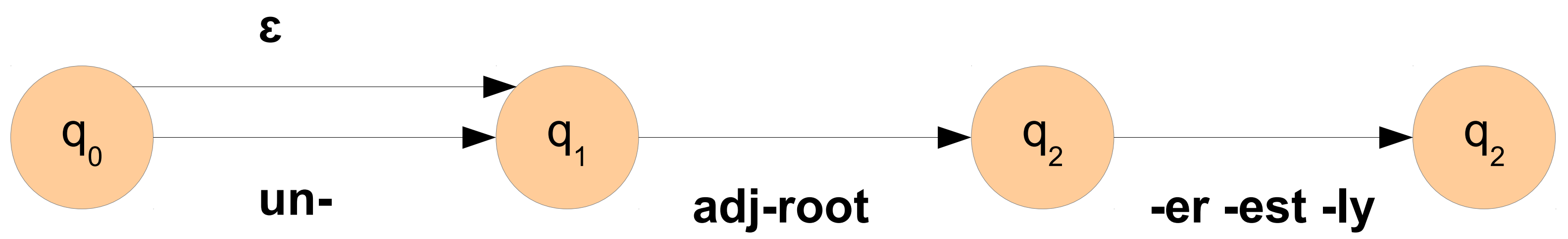
- FSA for English nominal inflection (same word category)



(Check example for verbal inflections in Jurafski & Martin book.)

Finite-State Lexicon

- FSA for derivational morphology (distinct word categories)



Correct adjectives:
- cooler
- unhappiest
- bigger

Incorrect adjectives:
- unbigger
- oranger
- smally

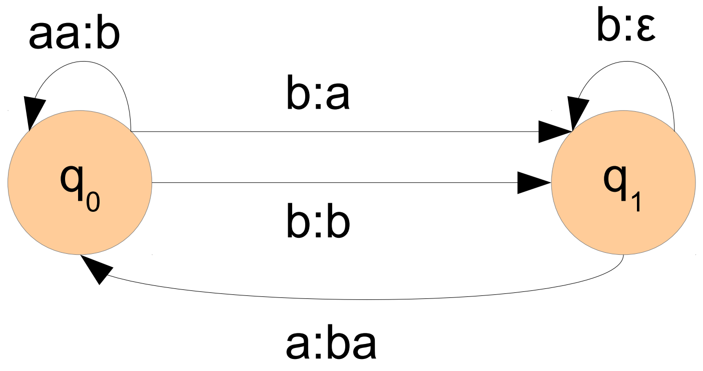
Solution: classes of roots (**adj-root**$_1$, **adj-root**$_2$, etc.)

Outline

- Regular Expressions
- Finite-State Automata
- Morphology
- Morphological parsing
- Finite-State Transducers

Finite-State Transducers (FST)

- FST is a type of FSA which maps between two sets of symbols.
- It is a two-tape automaton that recognizes or generates pairs of strings, one from each type.
- FST defines relations between sets of strings.



Finite-State Transducers for NLP

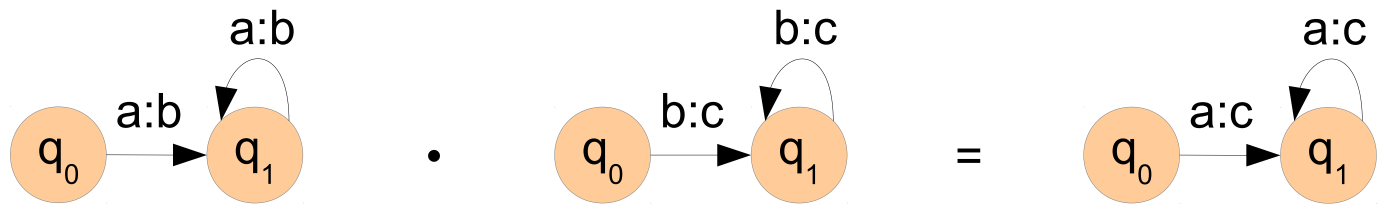
- FST as recognizer
  - Takes a pair of strings and accepts or rejects them
- FST as generator
  - Outputs a pair of strings for a language
- FST as translator
  - Reads a string and outputs another string
  - Morphological parsing: letters (input); morphemes (output)
- FST as relater
  - Computes relations between sets

Formalization of FST

- $Q = q_0 q_1 q_2 \ldots q_{N-1}$: finite set of N states
- Σ : finite input alphabet of symbols
- Δ : finite output alphabet of symbols
- q_0 : the start state
- F: the set of final states, $F \subseteq Q$
- $\delta(q, w)$: the transition function ($q \in Q$; input string $w \in \Sigma$) returns a set of new states $Q' \in Q$
- $\sigma(q, w)$: the output function ($q \in Q$; input string $w \in \Sigma$) returns a set of output strings $o \in \Delta$

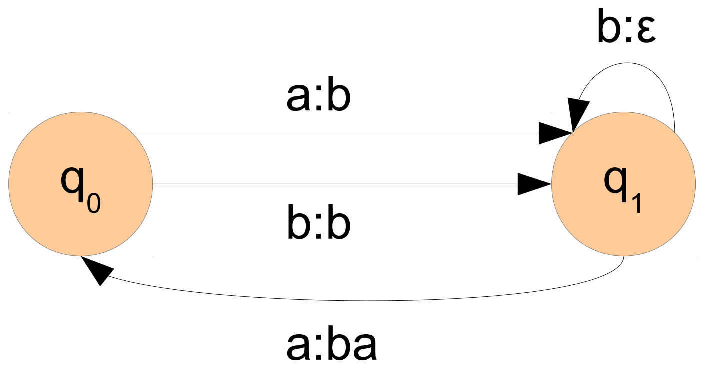
Regular Languages and Regular Relations

- FSA: Regular languages are sets of strings
- FST: Regular relations are sets of pairs of strings
  - Properties
    - Inversion: It switches the input and output labels
    - Composition:



Determinism for FST

- Not all FST can be determinized: They require search algorithms
- Sequential transducers: deterministic subtype of FST



- Subsequencial transducers
  - Generate and additional output string at the final states

Determinism for FST

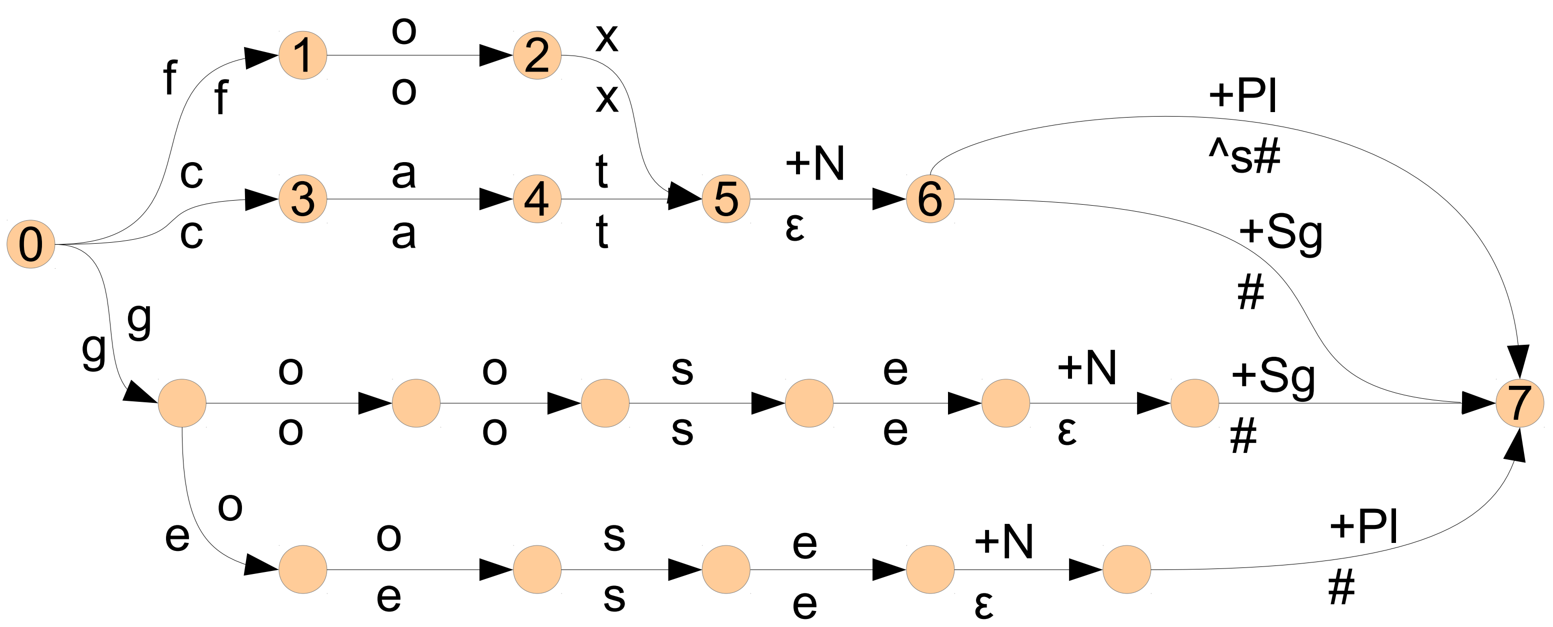
- Properties of sequential and subsequential transducers
  - Efficient
  - Linear in their input length
  - There are efficient algorithms for their determinization
  - But cannot handle ambiguity

FST for Morphological Parsing

- Two tapes
  - Upper (lexical) tape: input alphabet Σ
    - cat +N +Pl
  - Lower (surface) tape: output alphabet Δ
    - cats

FST for Morphological Parsing

- goose/geese: g:g o:e o:e s:s e:e
  - Feasible pairs (e.g., o:e) vs. default pairs (g:g)



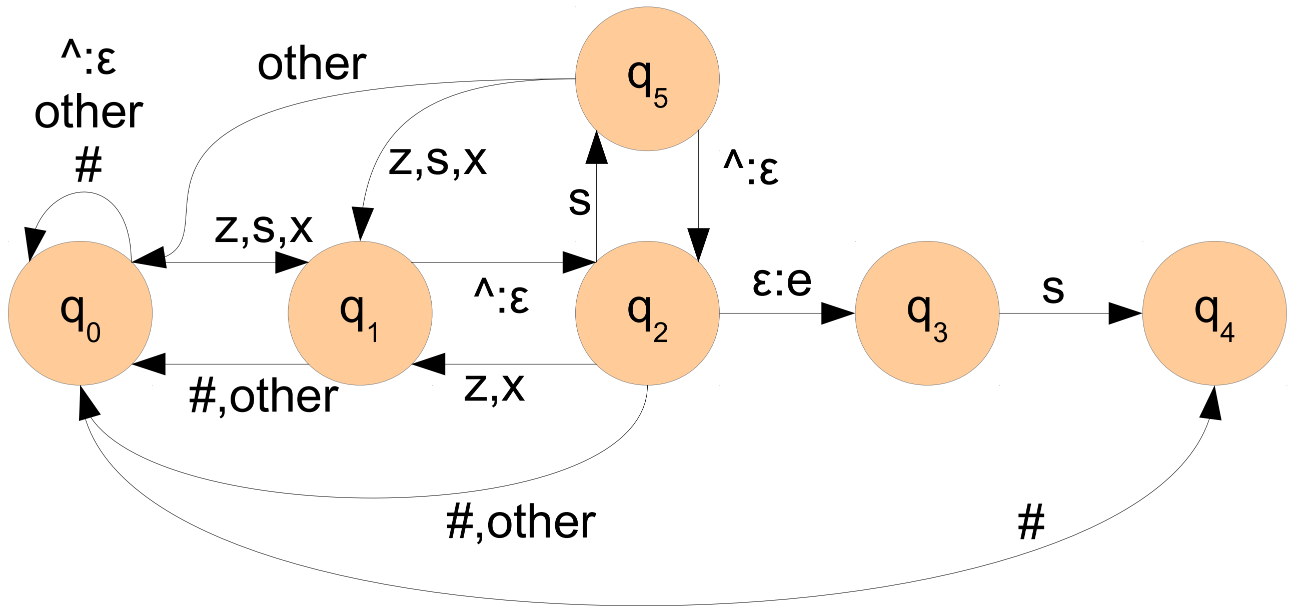
FST and Ortographical Rules

- Plural of „fox“ is „foxes“ not „foxs“

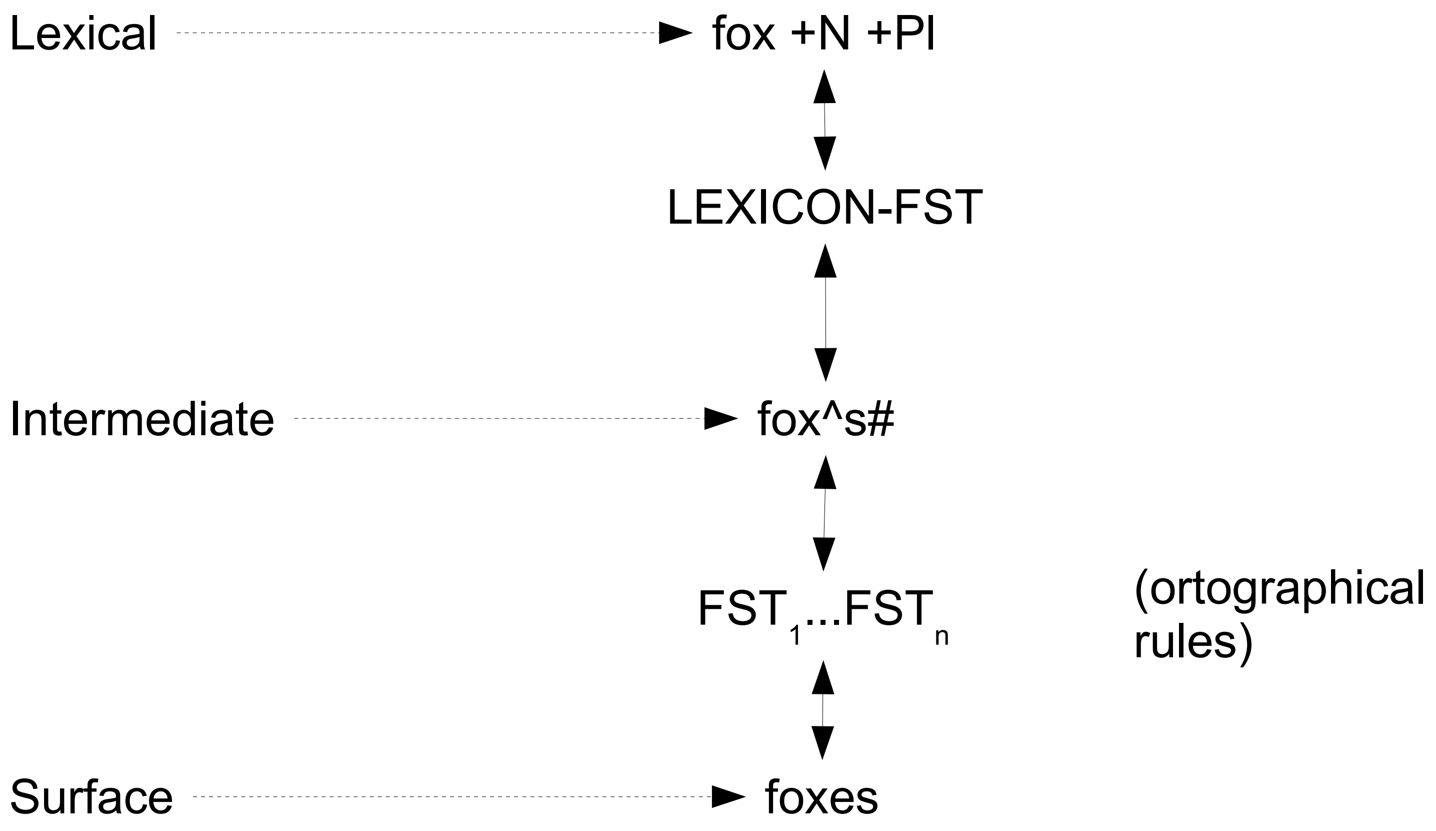
- Consonant double: beg/begging
- E deletion: make/making
- E insertion: watch/watches
- Y replacement: try/tries
- K insertion: panic/panicked

FST and Ortographical Rules

- Lexical: foxes +N +Pl
- Intermediate: fox^s#
- Surface: foxes



Combination of FST Lexicon and Rules

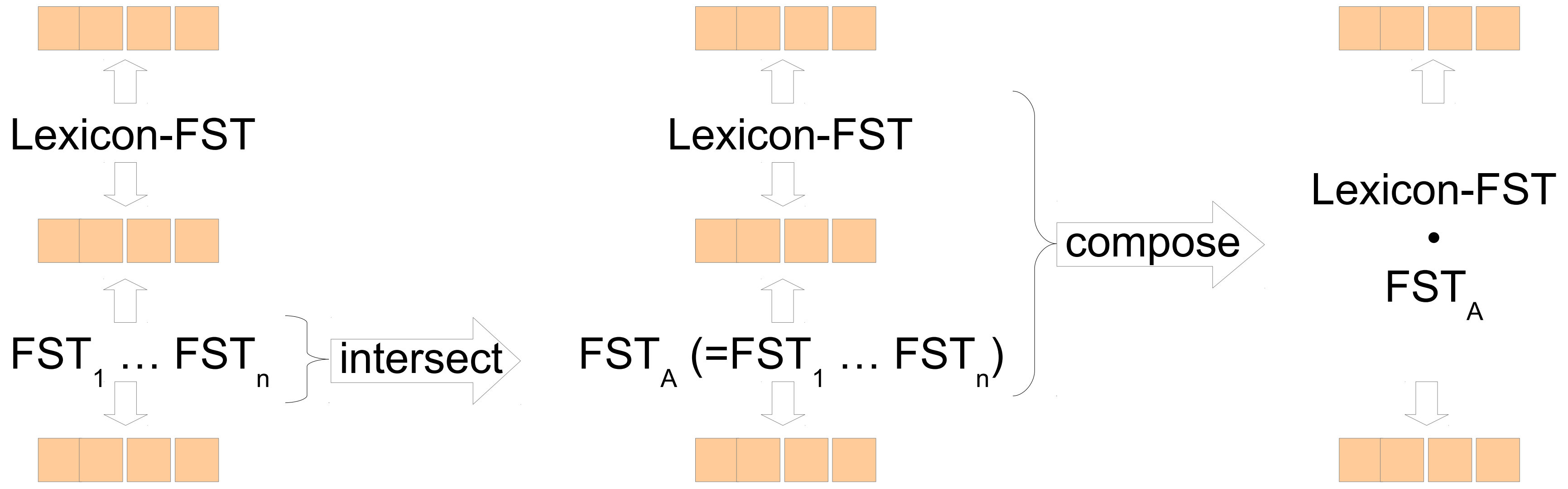


FST Lexicon and Rules

- Disambiguation
  - For some cases, it requires external evidences:
    - I saw two foxes yesterday. (fox +N +Pl)
    - That trickster foxes me every time! (fox +V +3SG)
  - But it can handle local ambiguity (intersection & composition)
    - „asses“ vs. „assess“

FST Lexicon and Rules

- Intersection & Composition



Porter Stemmer (Lexicon-Free FST)

- Popular for information retrieval and text categorization tasks
- It is based on a series of simple cascade rules
  - ATIONAL → ATE (relational → relate)
  - ING → ε (motoring → motor)
  - SSES → SS (grasses → grass)
- But it commits many errors:
  - ORGANIZATION → ORGAN
  - DOING → DOE

(http://tartarus.org/martin/PorterStemmer/)

Further reading

- Book Jurafski & Martin
  - Chapters 2 and 3 (until section 3.8)
- Regular expressions
  - http://www.regular-expressions.info/

Project update

- Deadline for deciding projects:
  - Friday, April 29th, 2016
- Short introduction of the projects (2/3 slides)
  - Next lecture (May 2nd, 2016)
  - What is it about? Which tools and data you plan to use?