Morphology
Finite State Transducers

Tanel Alumäe

English example

- *unladylike* – 3 morphemes
  un- ‘not’
  lady *‘(well behaved) female adult human’*
  -like ‘having the characteristics of’
- *technique* – 1 morpheme
- *dogs* – 2 morphemes
  dog
  -s a plural marker for nouns

Estonian example

- *põtradega* (‘with the deers’)
  - *põtra* a root of ‘*põder*’ (deer)
  - -*de* a plural marker
  - -ga a comitative case marker

- Recognizing that *põtradega* breaks down into the morphemes *põtra, de* and *ga* is called **morphological parsing**.
- Parsing (in general) means taking an input and producing some sort of structure for it

Turkish example

Turkish	English	Comment
Avrupa	Europe	
Avrupalı	of Europe / European	adjective (European)
Avrupalılaş	become European	(intransitive) verb root
Avrupalılaştır	to cause to become European / Europeanise	(transitive) verb root
Avrupalılaştırama	be unable to Europeanise	negated verb root
Avrupalılaştıramadık	unable to be Europeanised	participle
Avrupalılaştıramadık	one that is unable to be Europeanised	noun
Avrupalılaştıramadıklar	unable to be Europeanised ones	plural
Avrupalılaştıramadıklarımız	our unable to be Europeanised ones	possessive, 1st person plural
Avrupalılaştıramadıklarımızdan	of ours that were unable to be Europeanised	ablative case
Avrupalılaştıramadıklarımızdanmış	is reportedly of ours that were unable to be Europeanised	copula in inferential tense
Avrupalılaştıramadıklarımızdanmışsınız	you are reportedly of ours that were unable to be Europeanised	2nd person plural/formal
Avrupalılaştıramadıklarımızdanmışsınızcasına	as if you were reportedly of ours that were unable to be Europeanised	Adverb of equalization/possibility

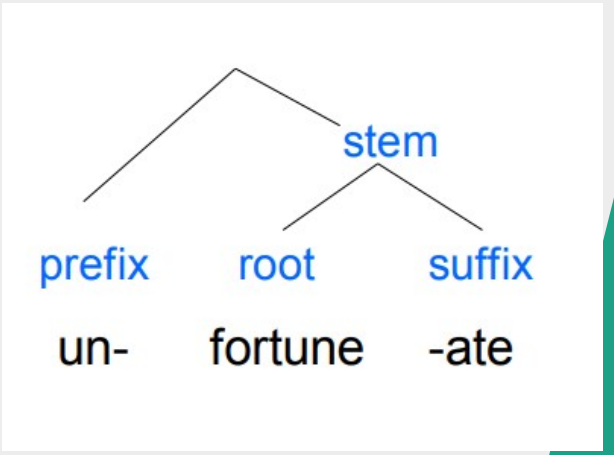
Why?

Morphological analysis is needed in many NLP applications, for example:

- Information retrieval:
  - search *vannivaht* (“bath foam”), retrieve documents containing *vannivahu* (“bath foam, genitive”)
  - Search *mouse*, retrieve documents containing *mice*
- Spell checking
  - Understand that the word “vannivahugagi” is a valid word, although it’s not in a dictionary
- Natural language understanding and generation in human-computer interaction systems:
  - User: *palun mulle kaks kilo **vahvleid** (please give me two kilos of **waffles**)*
    - The inflected word *vahvleid* has to be mapped to product name *vahvel*
  - Computer: ***vahvlid** on meil kahjuks otsas, soovite äkki **küpsiseid**?* (we are unfortunately out of waffles, maybe you want cookies?)
    - Computer might use a template: *<product_plural nominative> on meil kahjuks otsas, soovite äkki <replacement_product_plural_genitive>*
  -

Terminology

- Root – the portion of the word that
  - is common to a set of derived or inflected forms, if any, when all affixes are removed
  - is not further analyzable into meaningful elements
  - carries the principal portion of meaning of the words
- Stem - the root or roots of a word, together with any **derivational affixes**, to which inflectional affixes are added
- Affix - a bound morpheme that is joined before, after, or within a root or stem
  - Prefix, e.g. English ***un****buckle*
  - Suffix, e.g. English *eat****s***
  - Infix, e.g. Tagalog *hingi* "borrow" + ***um*** (marker for agent of an action) → *h****um****ingi* "borrower"
- Clitic - that has syntactic characteristics of a word, but depends phonologically on another word
  - English: *Hal's* (genetive marker)
  - Czech: *chceš****li*** ["(you) want-if" = "if you want"]
  - Estonian: *mina****gi*** (**even** I)



Inflectional *vs* Derivational

- Word Classes
  - **Parts of speech**: noun, verb, adjectives, etc.
  - Word class dictates how a word combines with morphemes to form new words
- Inflection
  - Variation in the form of a word, typically by means of an affix, that expresses a grammatical contrast.
    - Doesn’t change the word class
    - Usually produces a predictable, non-idiosyncratic change of meaning.
- Derivation:
  - The formation of a new word (often of a new class) or inflectable stem from another word or stem

Inflectional morphology

- Adds:
  - tense, number, person, mood, aspect
- Word class doesn’t change
- Word serves new grammatical role
- Examples
  - English: ***come*** is inflected for person and number:
    The pizza guy ***comes*** *at noon.*
  - Very common in Estonian (*tule**n**, tule**me**)*

Derivational morphology

- Nominalization (formation of nouns from other parts of speech, primarily verbs in English):
  - *computerization*
  - *appointee*
  - *killer*
  - *fuzziness*
- Formation of adjectives (primarily from nouns)
  - *computational*
  - *clueless*
  - *embraceable*
- Difficult cases:
  - *building* → from which sense of "*build*"?

Types of inflections

Inflected words can be formed in different ways:

- Affixation, or simply adding morphemes onto the word without changing the root
  - Prefixing, suffixing, infixing
- Reduplication, doubling all or part of a word to change its meaning
  - For example, Marshallese forms words meaning 'to wear X' by reduplicating the last consonant-vowel-consonant (CVC) sequence of a base, i.e. base+CVC:
    kagir 'belt'→ *kagirgir* 'to wear a belt' (kagir-gir)
    takin 'sock' → *takinkin* 'to wear socks' (takin-kin) (Moravsik 1978)
- Alternation, exchanging one sound for another in the root
  - Usually vowel sounds, as in the *ablaut* process found in Germanic strong verbs and the *umlaut* often found in nouns, among others)
    das *Haus* 'house' → *die Häuser* 'houses'
  - Also in Estonian
    käsi 'hand' → *käe* 'hand', singular genitive
- Suprasegmental variations, such as of stress, pitch or tone, where no sounds are added or changed but the intonation and relative strength of each sound is altered regularly.
  - Also in Estonian: *pall* – *palli* (2nd quantity degree, *välde*) – *palli* (3rd quantity degree)

Agglutinative languages

- In agglutinative languages
  - words may contain different morphemes to determine words' meanings
  - all of these morphemes (including stems and affixes) remain, in every aspect, unchanged after their unions,
  - resulting in generally more easily deducible word meanings
- Examples:
  - Estonian: *elevantidegagi* "even with elephants"
  - Turkish:
    uygarlaştıramadıklarımızdanmışsınızcasına
    uygar+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına
    "behaving as if you are among those whom we could not cause to become civilized"

Computational morphology

- English:
  - WORD STEM (+FEATURES)*
    cats cat +N +Pl
    cat cat +N +Sg
    cities city +N +Pl
    geese goose +N +Pl
    ducks duck +N +Pl or
    duck +V +3Sg
    going go +V +Pres-Part
    caught catch +V +Past-Part or
    catch +V +Past

Computational morphology

- Estonian

  loen
  luge+n //_V_ n, //

  karu
  karu+0 //_S_ sg g, sg n, sg p, //

  talle
  tall+0 //_S_ adt, sg p, // (‘*ta vaatas väikest talle*’)
  tall+0 //_S_ sg g, // (‘*lammas sai väikse talle*’)
  tall+e //_S_ pl p, // (‘*nägin suuri talle’)*
  tema+lle //_P_ sg all, // (‘*helistan talle’)*

- Try: http://www.filosoft.ee/html_morf_et/

Computation morphology: approaches

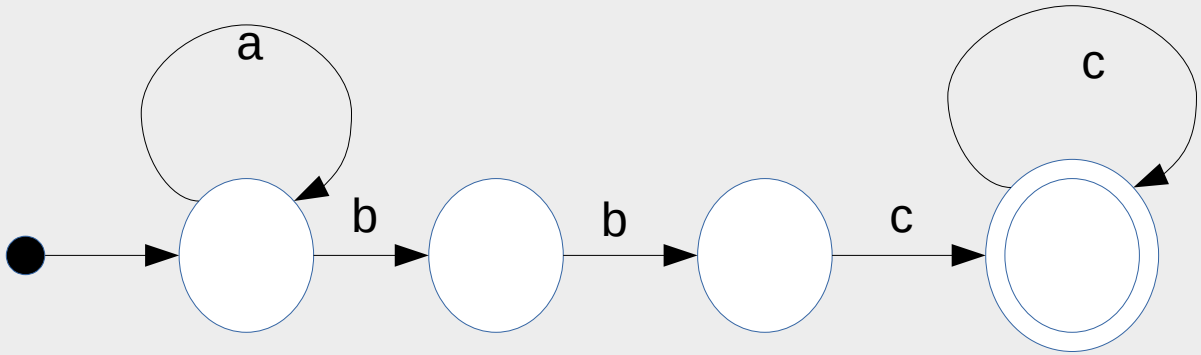
- Lexicon: for all *word forms*, define its morphological analysis
  uba = uba+0 //_S_ sg n, sg p, //
  oa = uba+0 //_S_ sg g, //
  ...
  - Problem: in many languages, new words can be formed spontaneously, using derivation (*oalik* “bean-like”) and compounding (*oakott* “bean bag”)
  - Inefficient, especially for morphologically complex languages like Estonian
- Rules
- Lexicon and rules:
  - Group words into classes where all class members have similar **morphological behaviour**
  - Build inflection rules for each class
  - Estonian:
    - Class 16S: *ratsu – ratsu – ratsut – ratsusse*
    - Includes words like *kõne, male, klade, auto, kiisu, lauljanna,*

How to build rules?

- One powerful tool for building morphological rules is the framework of finite state transducers
- Finite state transducers define a *relation* between sets of strings
  - E.g.:
    - Input: *cats*
    - Output: *cat +N +PL*

Finite state automata

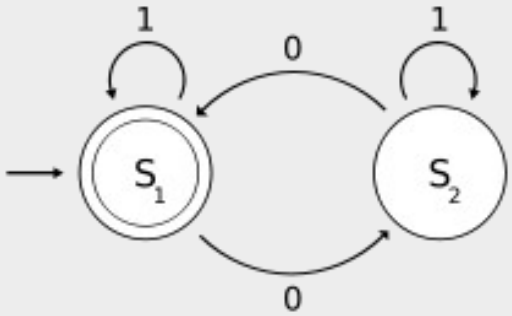
- Recall regular expressions. e.g.: */a*bbc+/*
- Finite state automata can be viewed as a formal generalization of regular expressions



Finite state automaton: definition

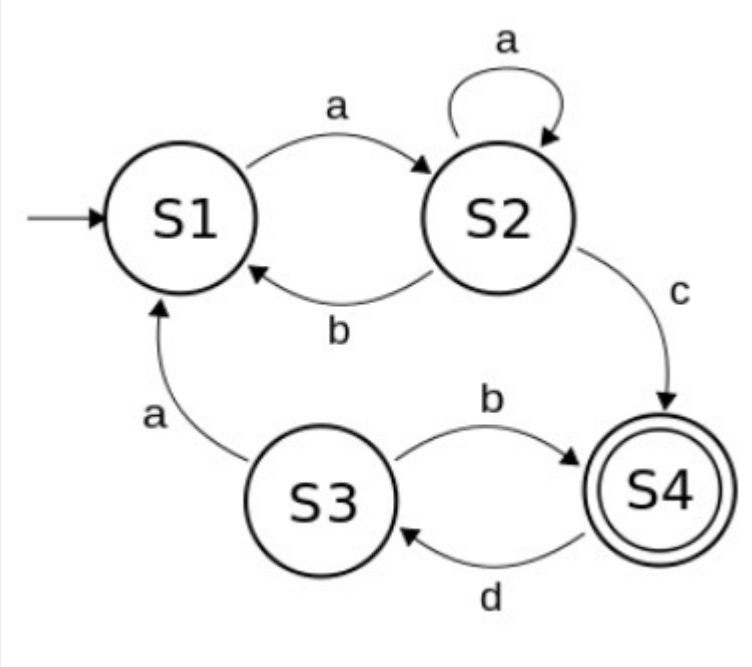
Deterministic finite state automaton has the following components:

- The set of states: Q
- A finite alphabet: Σ
- A start state
- A set of accept/final states
- A transition function δ that maps QxΣ to Q



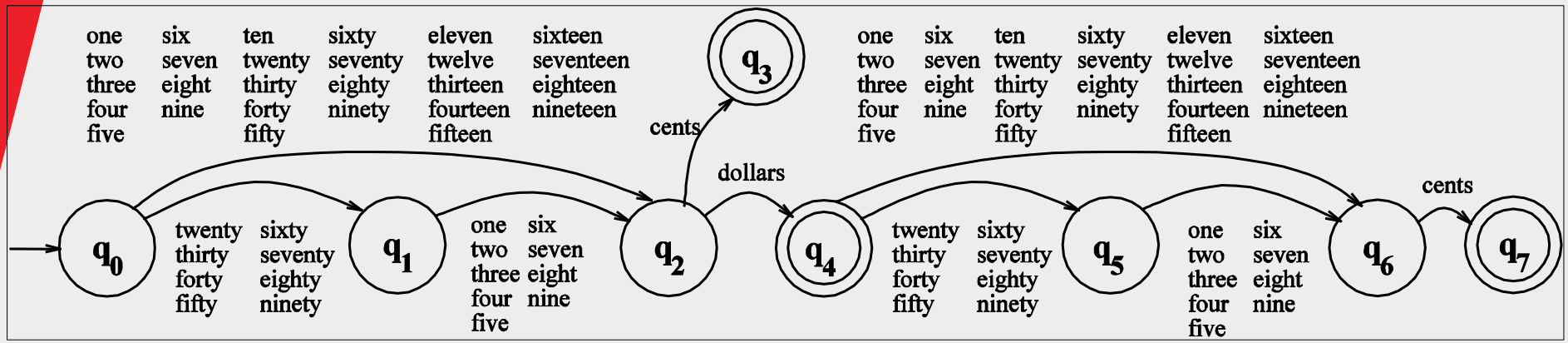
Quiz

- Which string **cannot** be generated by the finite state automaton:



- abac
- abacdaac
- aaaaac
- aaaaaacd

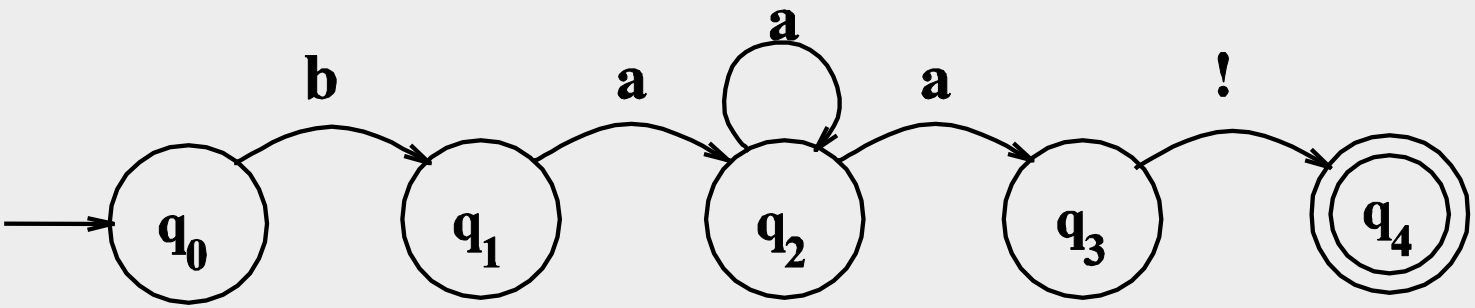
FSA example: dollars and cents



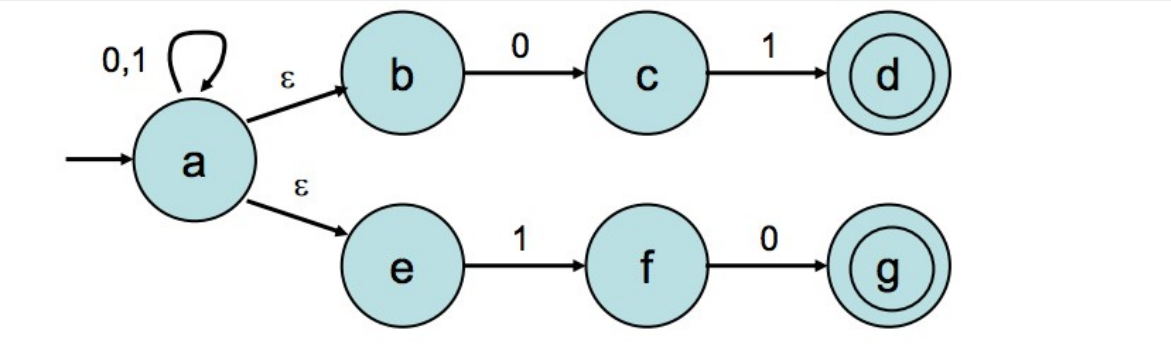
- Accepts the following sentences:
  - two cents
  - three dollars
  - twenty dollars twenty one cents
  - one dollars ten cents

Non-deterministic FSA

- Unlike deterministic FSA, non-deterministic FSAs allow multiple transition functions in the same state for the same symbol



- Additionally, non-deterministic FSA can use null transitions, which are indicated by ε
- Null transitions allow the machine to jump from one state to another without having to read a symbol

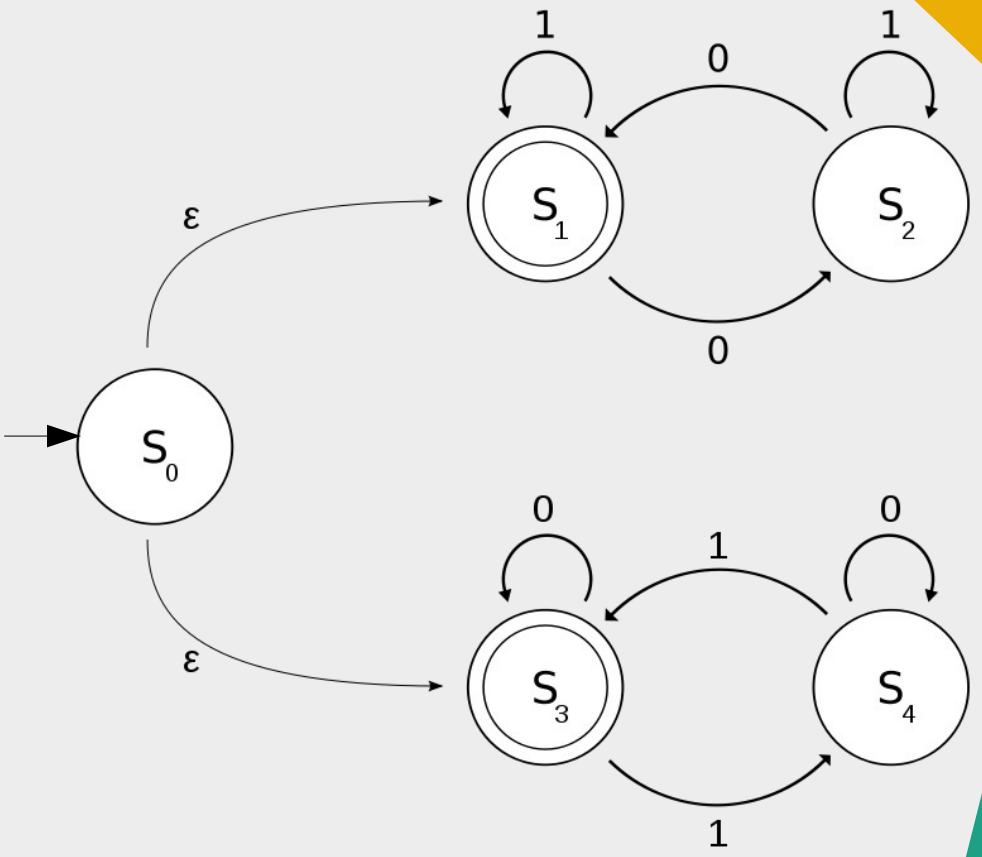


Recognition

- Recognition is the process of determining if a string should be accepted by a machine
- Or… it’s the process of determining if a string is in the language we’re defining with the machine

Quiz

- Which string cannot be generated by the FSA on the right?
  - 1011101
  - 0
  - 01001
  - 1000

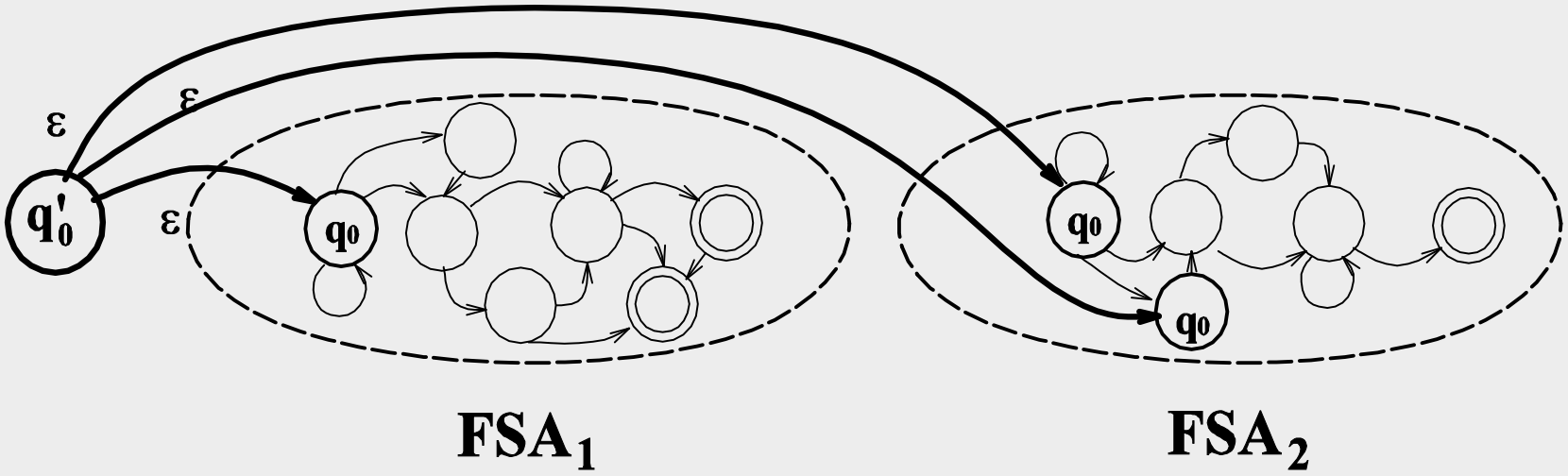


FSA operations

- Various operations can be applied to FSAs:
  - Concatenation
  - Union
  - Negation
  - Intersection
  - Closure
    - Kleene star (*)
    - +

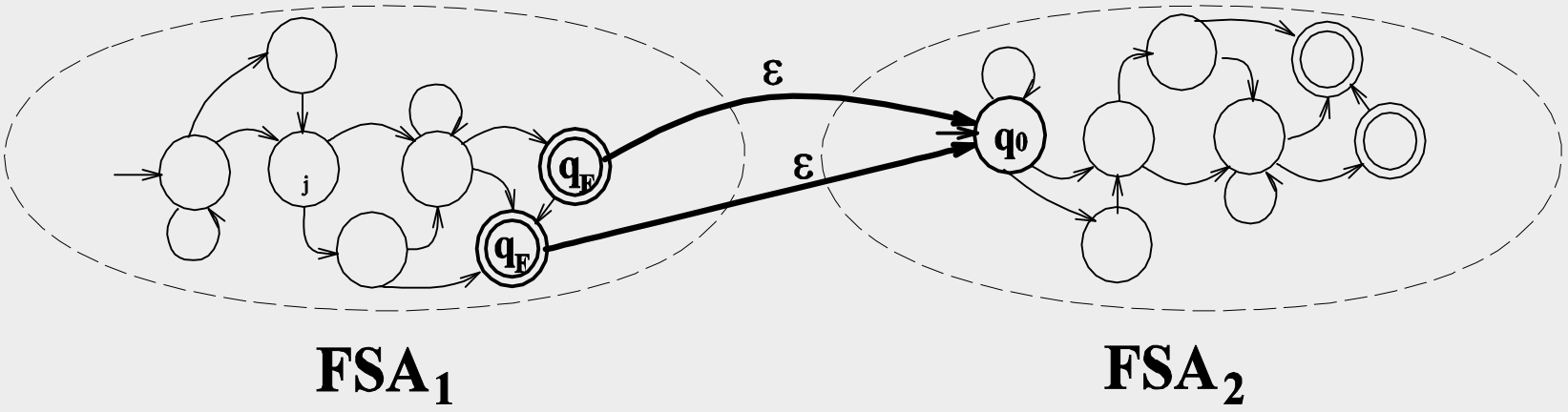
FSA Union

Accept a string in either of two languages



FSA Concatenation

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

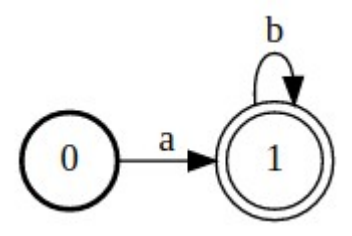


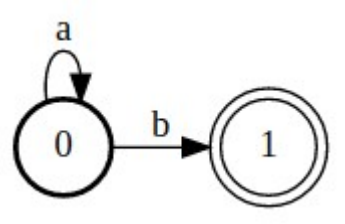
Negation

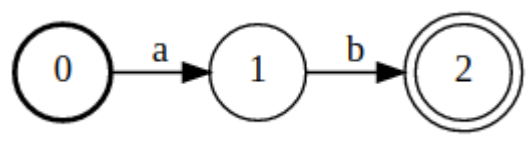
- Construct a machine M2 to accept all strings not accepted by machine M1 and reject all the strings accepted by M1
- Invert all the accept and not accept states in M1

Intersection

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



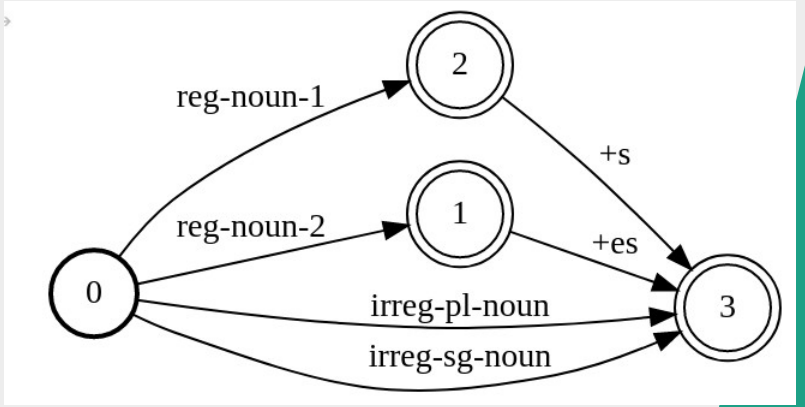




FSAs and computational morphology

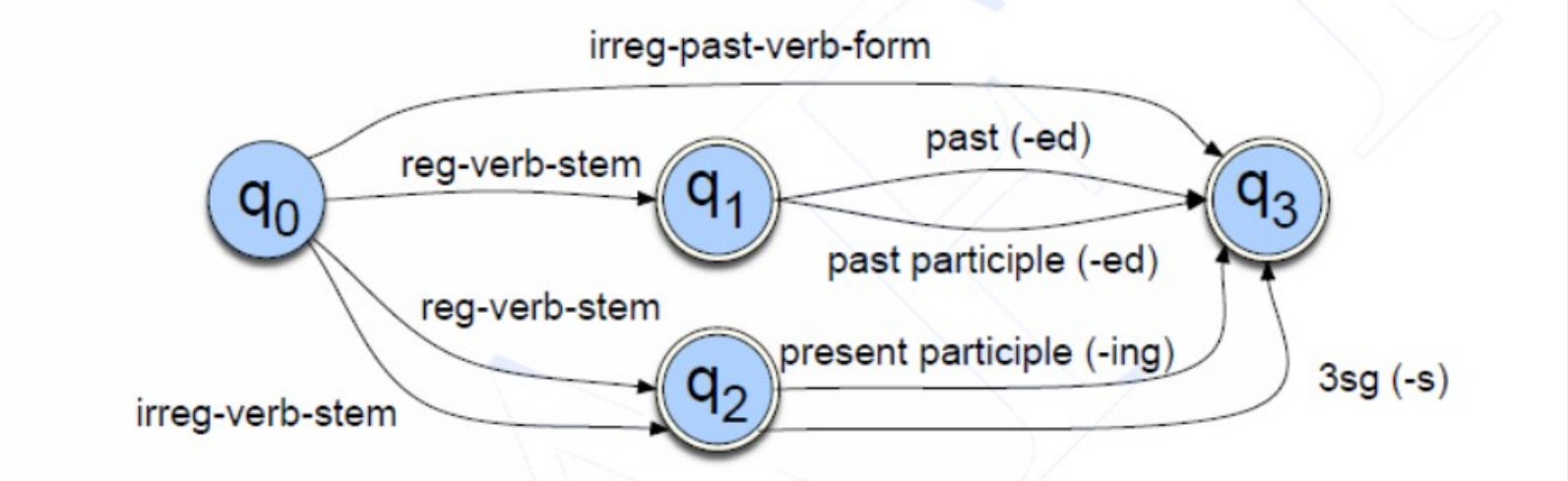
- FSA can be used formally describe all words (or a subset of words) in the language
  - E.g., an incomplete FSA for describing all nouns (singular and plural) in English

reg-noun1 (-s)	reg-noun2 (-es)	irreg-pl-noun	irreg-sg-noun
dog	*fox*	*geese*	*goose*
cat	*ostrich*	*sheep*	*sheep*
aardvark		*mice*	*mouse*



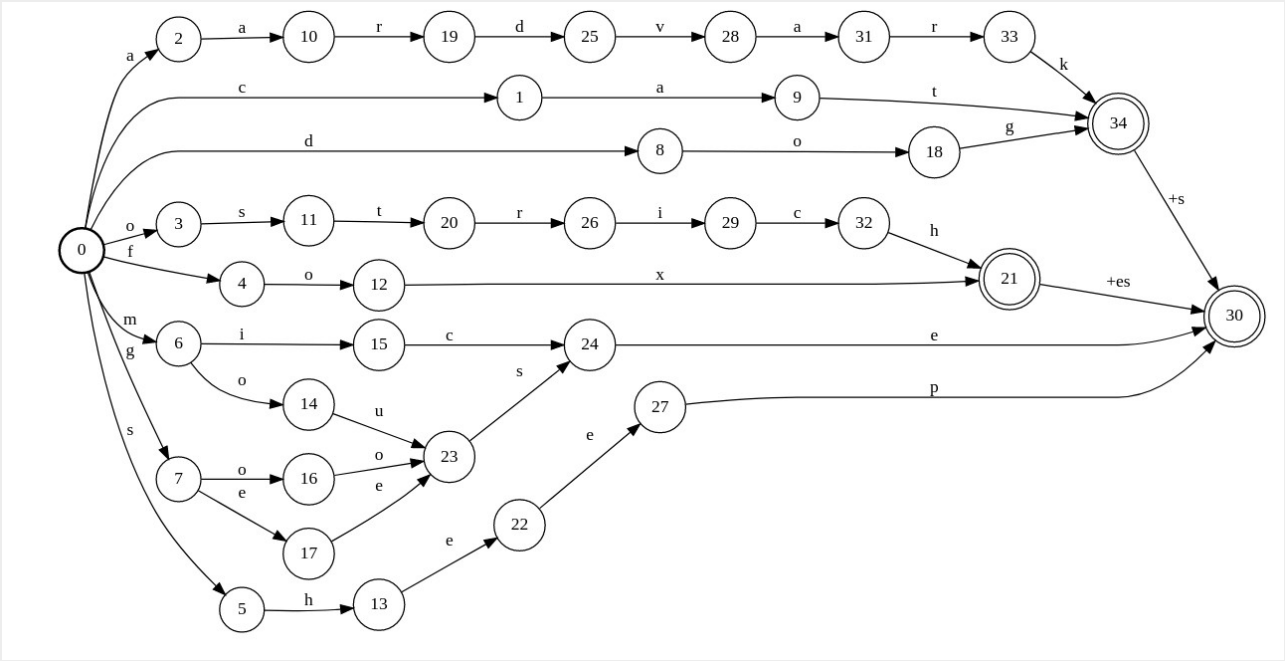
FSA and computation morphology

- Example 2: English verbal inflection
  - reg-verb-stem: *walk, fry, talk, impeach*
  - irreg-verb-stem: *cut, speak, sing*
  - irreg-past-verb: *caught, ate, eaten, sang*
  - past: *-ed*
  - past-part: *-ed*
  - pres-part: *-ing*
  - 3sg: *-s*



FSA and computational morphology

- Determining whether an input string of letters makes up a legitimate English word or not.
- We do this by taking the FSAs and plugging in each “sublexicon” into the FSA.
- That is, we expand each arc (e.g., the **reg-noun-1** arc) with all the morphemes that make up the set of **reg-noun-1**.
- The resulting FSA is defined at the level of the individual letter
- It only shows the distinction between recognizing regular and irregular forms

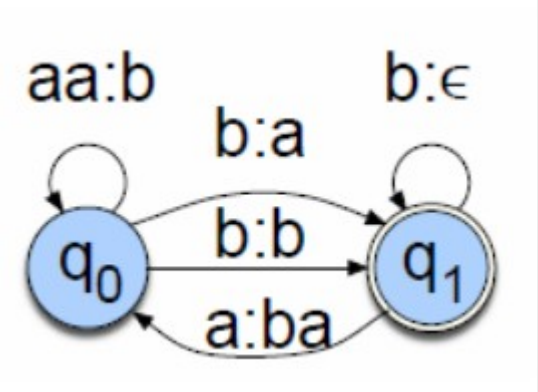


Morphological recognition vs morphological analysis

- Recognition is usually not quite what we need
- Usually if we find some string in the language we need to find the structure in it (morphological parsing)
- Or we have some structure and we want to produce a surface form (production/generation)
- Example:
  - From *cats* to *cat +N +PL* and back
  - From *loen* to *luge+n //_V_ n, //* and back

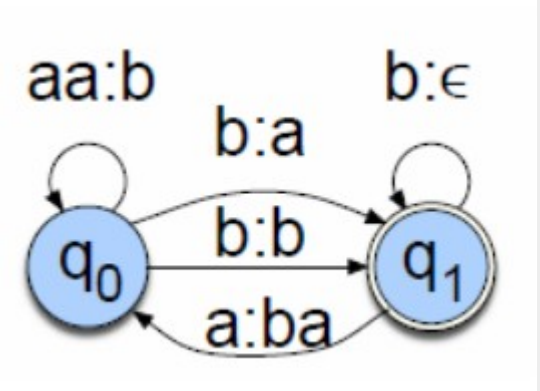
Finite State Transducers

- A finite-state transducer or FST is a type of finite automaton which maps between two sets of symbols
- This can be done by labeling each arc in the finite-state machine with two symbol strings
- Another way of looking at an FST is as a machine that generates an output string for each input string that it accepts
- Example:
  b → a
  aab → ba
  bbbbbbbb → a



Ambiguity

- Recall that in **non-deterministic** recognition multiple paths through a machine may lead to an accept state.
  - It didn't matter which path was actually traversed
- In FSTs the path to an accept state does matter since different paths represent different parses and different outputs will result from them
- That's OK for morphological parsing, since a word form can have multiple analyses (e.g., *read* is a present or past verb, or a noun)
- Example:
  b → *a*
  b → *b*

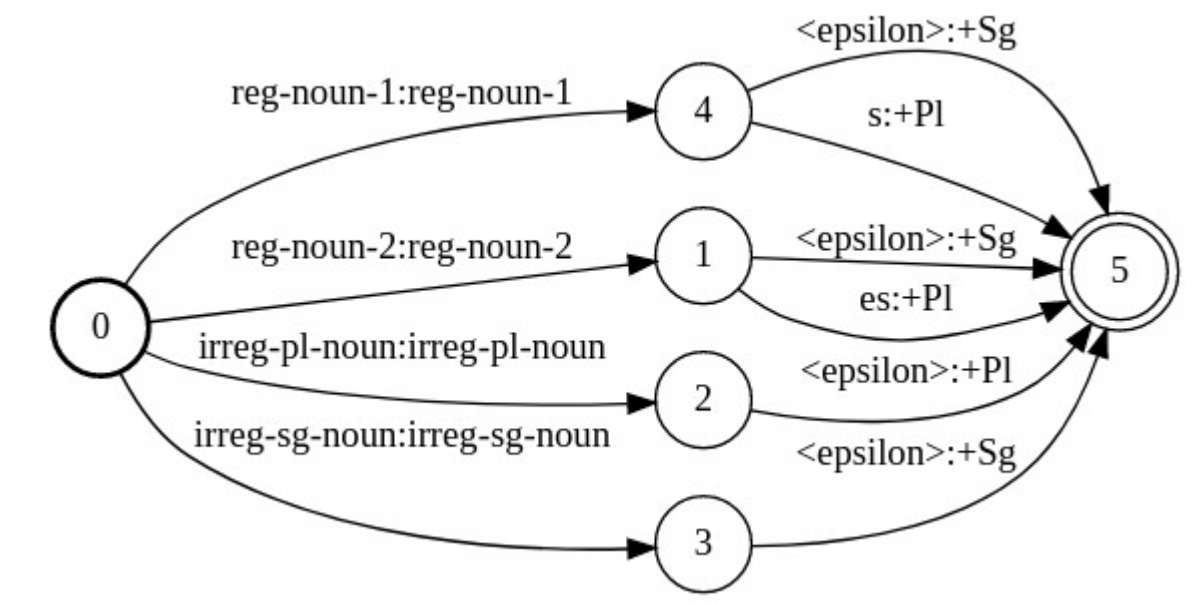


FST operations

- FST can be inverted and composed (very useful in practice)
  - Inversion: The inversion of a transducer T (T^{-1}) switches the input and output labels. Thus if T maps from the input alphabet I to the output alphabet O, T^{-1} maps from O to I
    - T: $b \rightarrow a$, $aab \rightarrow ba$, $abbbbbbbb \rightarrow a$
    - T^{-1}: $a \rightarrow b$, $ba \rightarrow aab$, $a \rightarrow abbbbbbb$
  - Composition: If $T1$ is a transducer from $I1$ to $O1$ and $T2$ a transducer from $O1$ to $O2$, then $T1 \circ T2$ maps from $I1$ to $O2$

FSTs for morphological processing

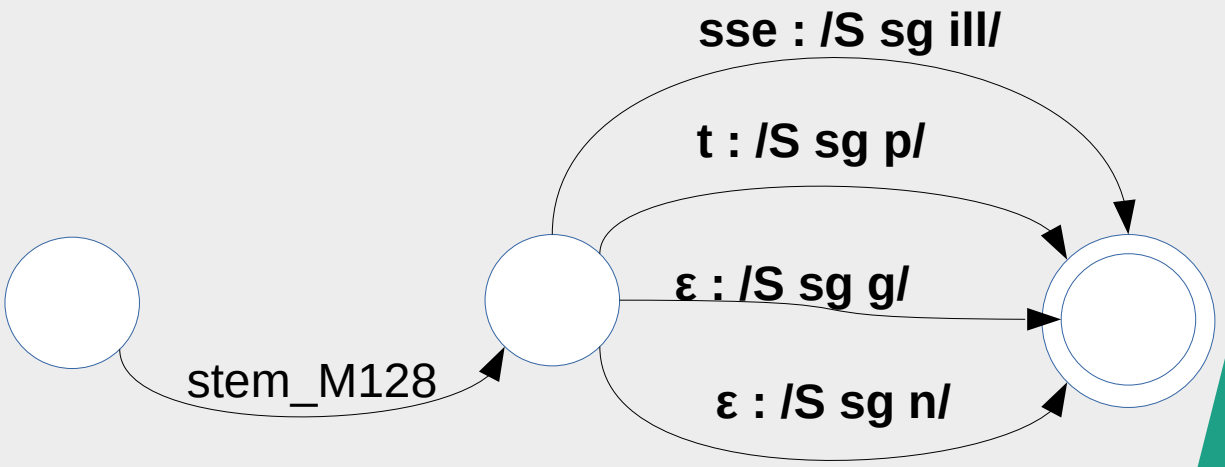
- FST that maps plural nouns (like *cats*) to a format *stem +Pl* (like *cat +Pl*) and singular nouns to a format *stem +Sg*
  - *reg-noun: fox, cat, aardvark*
  - *irreg-pl-noun: geese:goose, sheep, mice:mouse*
  - *irreg-sg-noun: goose, sheep, mouse*
- Arc labels: <input>:<output>
- Label <epsilon> stands for empty string



Estonian example

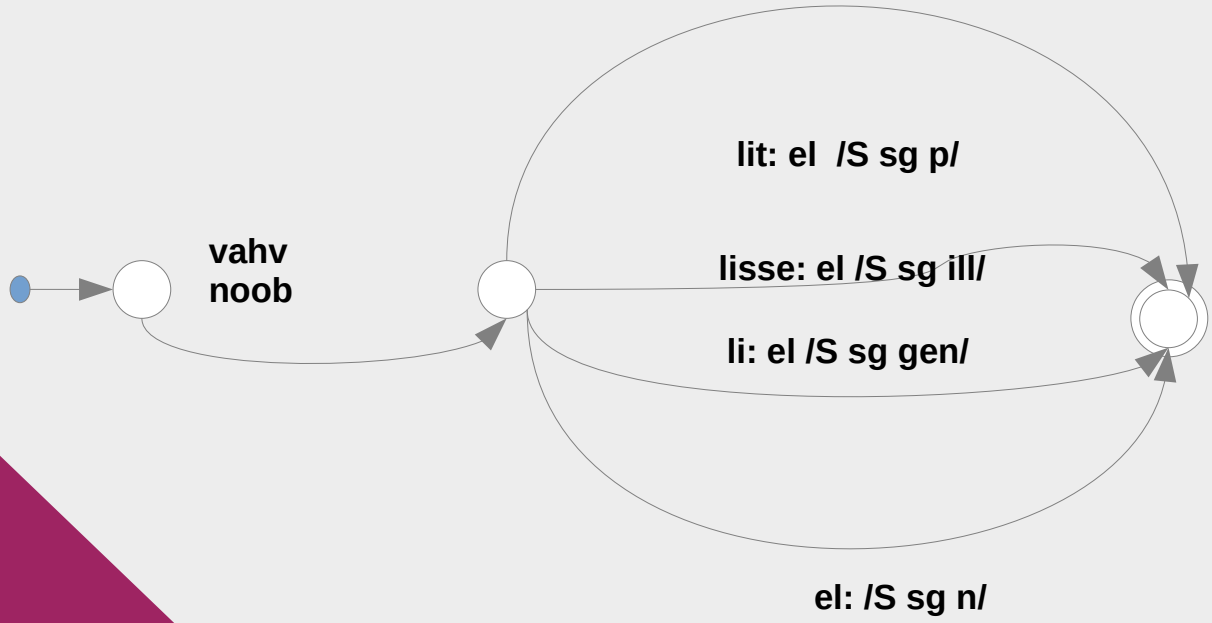
- Recall:
  - Class M128: *ratsu – ratsu – ratsut – ratsusse*
  - Includes words like *kõne, male, klade, auto, kiisu, lauljanna*

stem_M128
ratsu
kõne
male
...



More complex Estonian example

- What if word's stem also changes, e.g. *vahvel – vahvli - vahvlit – vahvlisse*?
- According to EKI (Institute of Estonian Language), *vahvel* inflects as *number, noobel*
- Goal: *vahvlisse → vahvel /S sg ill/*

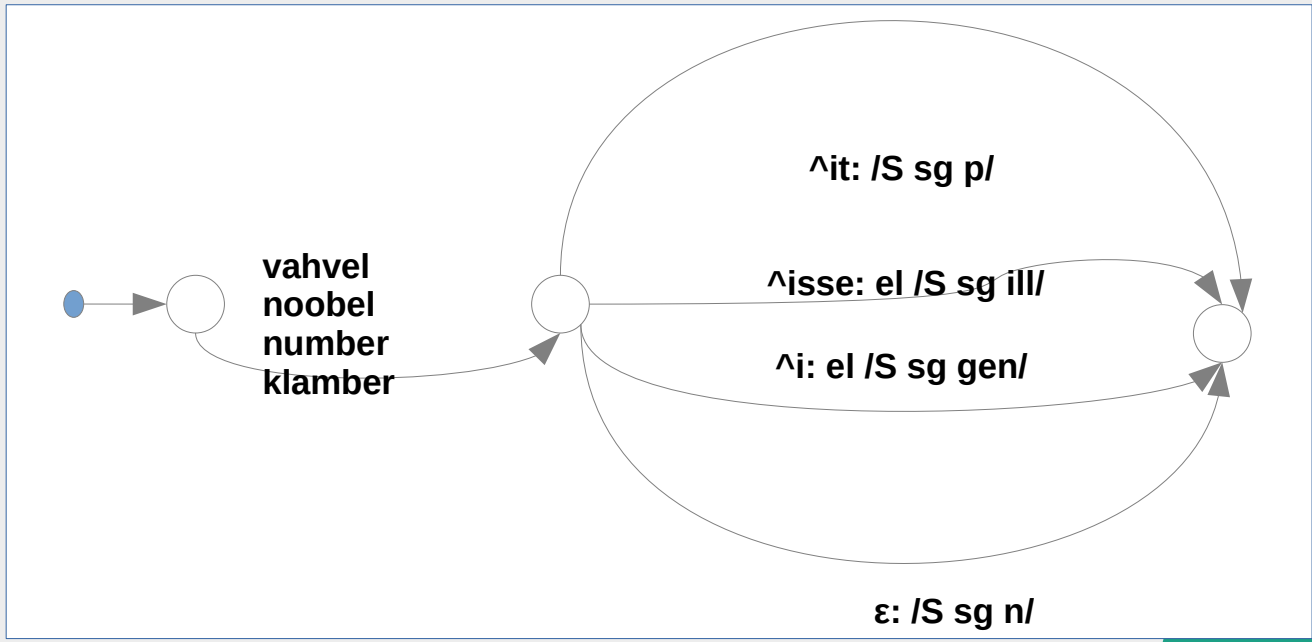


Multistage FST

- The problem with previous example: *vahvel* inflects as *number*, however, they differ in the *l/r*
- Naive solution: create different FSTs for words [*vahvel, noobel*] and [*number, klamber*]
- More compact solution: use one FST to transform *vahvlisse → vahvel^isse* and a second FST to “fix” *vahvel^isse → vahvel^isse → vahvlisse* and *number^isse → numbrisse*
- Finally, **compose** the two FSTs
- The 1st FST can be implemented using context-dependent rewrite rules

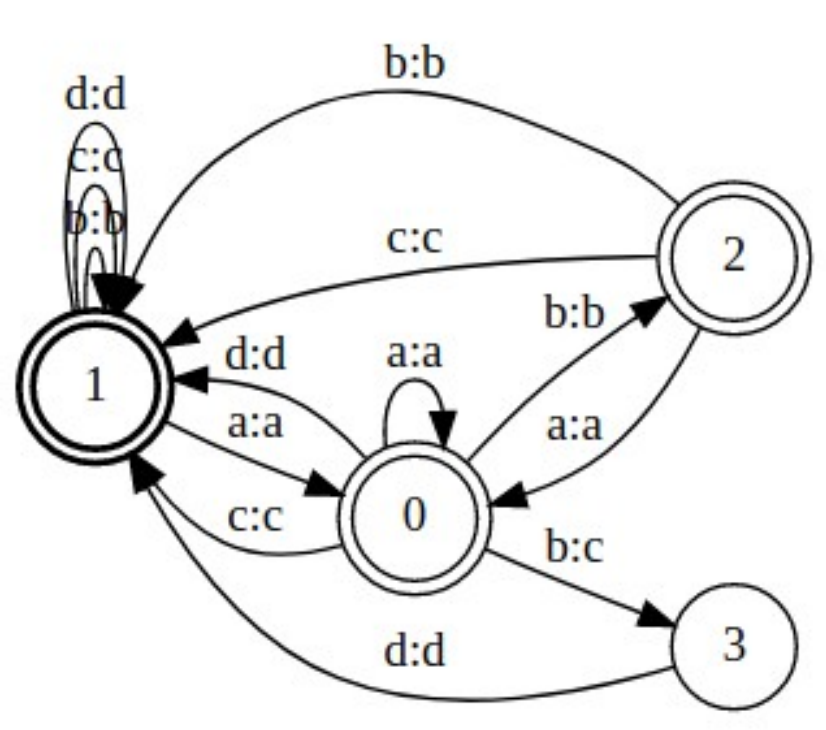
FST that changes
*vahv**li** → vahv**el^i***
*numb**rit** → numb**er^it***

i.e.:
*l → el^*
*r → er^*
when followed
by a suffix
i, it, isse, iga, ...



Context-dependent rewriting with FSTs

- Context-dependent rewriting is a very common operation
- It’s easy to define and understand but it’s representation as a FST is quite complicated
- Don’t worry about this, as most FST toolkits do it for you
- For example, the FST below transforms *b* to *c* when *b* occurs between *a* and *d*
  - For example: **abbabdc → abbacdc**
- The standard notation for this is: **b → c/a___d**



FST use cases

- FSTs can be used for other NLP tasks:
  - Converting numbers to words and back
    128 → *one hundred twenty eight*
    one hundred twenty eight → 128
  - Expanding dates, times, street names, etc to words and back
  - Expanding units (*mm → millimeters*)
  - In the Estonian speech recognition system, FSTs are used for converting spoken words to numbers, e.g.:
    ”... saja ühele protsendile..." => "*... 101-le protsendile...*"
  -

Questions?