(Overview of) Natural Language Processing Lecture 2: Morphology and finite state techniques

Paula Buttery (Materials by Ann Copestake)

Computer Laboratory
University of Cambridge

October 2019

Outline of today's lecture

Lecture 2: Morphology and finite state techniques
- A brief introduction to morphology
- Using morphology in NLP
- Aspects of morphological processing
- Finite state techniques
- More applications for finite state techniques

Morphology is the study of word structure

We need some vocabulary to talk about the structure:

- morpheme: a minimal information carrying unit
- affix: morpheme which only occurs in conjunction with other morphemes (affixes are bound morphemes)
- words made up of stem and zero or more affixes. e.g. *dog+s*
- compounds have more than one stem. e.g. *book+shop+s*
- stems are usually free morphemes (meaning they can exist alone)
- Note that *slither*, *slide*, *slip* etc have somewhat similar meanings, but *sl-* not a morpheme.

Affixes comes in various forms

- suffix: *dog+s*, *truth+ful*
- prefix: *un+wise*
- infix: (maybe) *abso-bloody-lutely*
- circumfix: not in English
  German *ge+kauf+t* (stem *kauf*, affix *ge_t*)

Listed in order of frequency across languages

Inflectional morphemes carry grammatical information

- Inflectional morphemes can tell us about tense, aspect, number, person, gender, case...
- e.g., plural suffix *+s*, past participle *+ed*
- all the inflections of a stem are often referred to as a paradigm

Derivational morphemes change the meaning

- e.g., *un-*, *re-*, *anti-*, *-ism*, *-ist* ...
- broad range of semantic possibilities, may change part of speech: *help* $\rightarrow$ *helper*
- indefinite combinations:
  antiantidisestablishmentarianism
  anti-anti-dis-establish-ment-arian-ism

Languages have different typical word structures

- isolating languages: low number of morphemes per word (e.g. Yoruba)
- synthetic languages: high number of morphemes per word
  - agglutinative: the language has a large number of affixes each carrying one piece of linguistic information (e.g. Turkish)
  - inflected: a single affix carries multiple pieces of linguistic information (e.g. French)

What type of language is English?

English is an analytic language

English is considered to be analytic:

- very little inflectional morphology
- relies on word order instead
- and has lots of helper words (articles and prepositions)
- but not an isolating language because has derivational morphology

English is an analytic language

English has a mix of morphological features:

- suffixes for inflectional morphology
- but also has inflection through sound changes:
  - *sing*, *sang*, *sung*
  - *ring*, *rang*, *rung*
  - BUT: *ping*, *pinged*, *pinged*
  - the pattern is no longer productive but the other inflectional affixes are
- and what about:
  - *go*, *went*, *gone*
  - *good*, *better*, *best*
- uses both prefixes and suffixes for derivational morphology
- but also has zero-derivations: *tango*, *waltz*

Internal structure and ambiguity

Morpheme ambiguity: stems and affixes may be individually ambiguous: e.g. *paint* (noun or verb), *+s* (plural or 3persg-verb)
Structural ambiguity: e.g., *shorts* or *short -s*
blackberry blueberry strawberry cranberry
unionised could be *union -ise -ed* or *un- ion -ise -ed*
Bracketing: *un- ion -ise -ed*

- *un- ion* is not a possible form, so not *((un- ion) -ise) -ed*
- *un-* is ambiguous:
  - with verbs: means 'reversal' (e.g., *untie*)
  - with adjectives: means 'not' (e.g., *unwise, unsurprised*)
- therefore *(un- ((ion -ise) -ed))*

Using morphological processing in NLP

- compiling a full-form lexicon
- stemming for IR (not linguistic stem)
- lemmatization (often inflections only): finding stems and affixes as a precursor to parsing
  morphosyntax: interaction between morphology and syntax
- generation
  Morphological processing may be bidirectional: i.e., parsing and generation.

```
party + PLURAL <-> parties
sleep + PAST_VERB <-> slept
```

Spelling rules

- English morphology is essentially concatenative
- irregular morphology — inflectional forms have to be listed
- regular phonological and spelling changes associated with affixation, e.g.
  - *-s* is pronounced differently with stem ending in *s*, *x* or *z*
  - spelling reflects this with the addition of an *e* (*boxes* etc)

  morphophonology
- in English, description is independent of particular stems/affixes

e-insertion

e.g. *box^s* to *boxes*

$$\varepsilon \rightarrow \text{e} / \left\{ \begin{array}{c} \text{s} \\ \text{x} \\ \text{z} \end{array} \right\} \hat{} \_ \, s$$

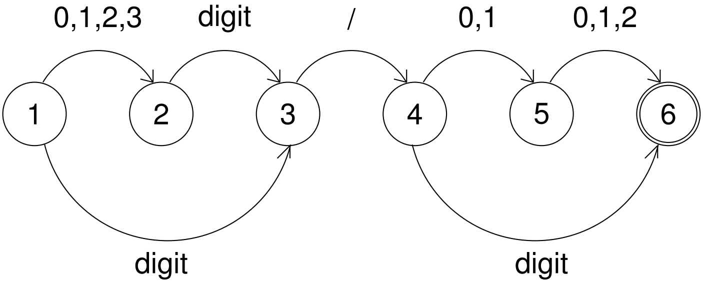
- map 'underlying' form to surface form
- mapping is left of the slash, context to the right
- notation:

_	position of mapping
ε	empty string
^	affix boundary — stem ^ affix

- same rule for plural and 3sg verb
- formalisable/implementable as a finite state transducer

Finite state automata for recognition

day/month pairs:



- non-deterministic — after input of ‘2’, in state 2 and state 3.
- double circle indicates accept state
- accepts e.g., 11/3 and 3/12
- also accepts 37/00 — overgeneration

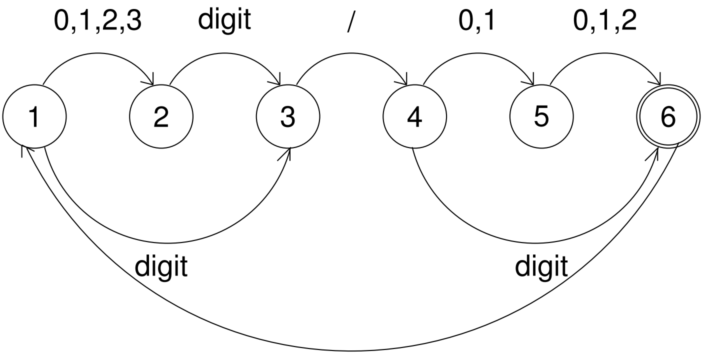
Reminder: Finite-State Automata

FSA are defined as $M = (\mathcal{Q}, \Sigma, \Delta, s, \mathcal{F})$ where:

- $\mathcal{Q} = \{q_0, q_1, q_2 ...\}$ is a finite set of states.
- Σ is the alphabet: a finite set of transition symbols.
- $\Delta \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$ is a function $\mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ which we write as δ. Given $q \in \mathcal{Q}$ and $i \in \Sigma$ then $\delta(q, i)$ returns a new state $q' \in \mathcal{Q}$
- s is a starting state
- $\mathcal{F}$ is the set of all end states

Recursive FSA

comma-separated list of day/month pairs:



- list of indefinite length
- e.g., 11/3, 5/6, 12/04

e-insertion

e.g. *box^s* to *boxes*

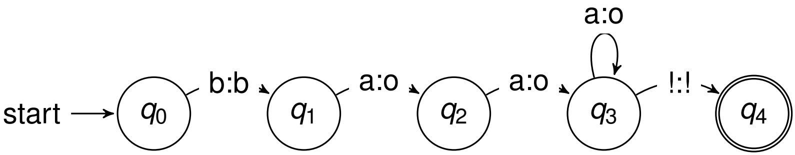
$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{array}{c} \mathrm{s} \\ \mathrm{x} \\ \mathrm{z} \end{array} \right\} \hat{}\ \_\ s$$

- map 'underlying' form to surface form
- mapping is left of the slash, context to the right
- notation:

_	position of mapping
ε	empty string
^	affix boundary — stem ^ affix

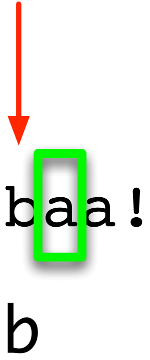
Finite State Transducers for Morphology

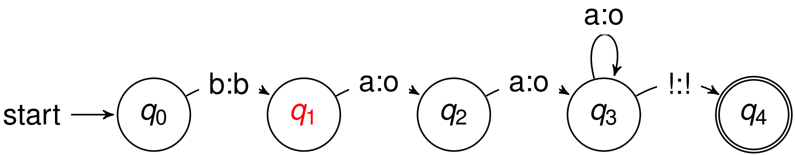
We will be attempting to map between a word and its structure and to do this we will need an augmentation to the FSA; something called a *Finite state transducer* (FST).



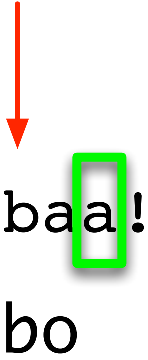
- FST are used to map between representations.
- You can think of a FST as being FSA which produces two sequences for any given path through the states;
- Or alternatively as an FSA which maps one string into another.

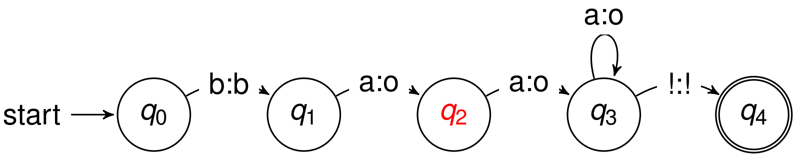
The operation of a FST



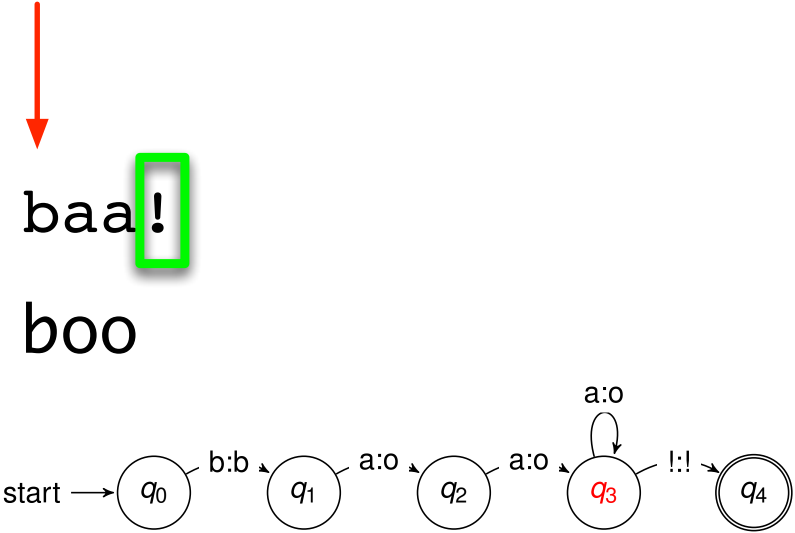


The operation of a FST





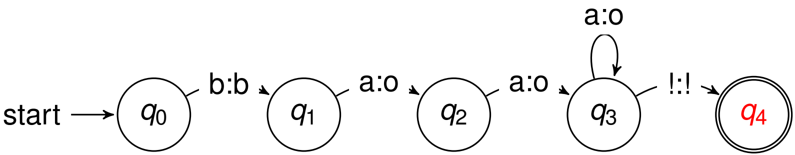
The operation of a FST



The operation of a FST

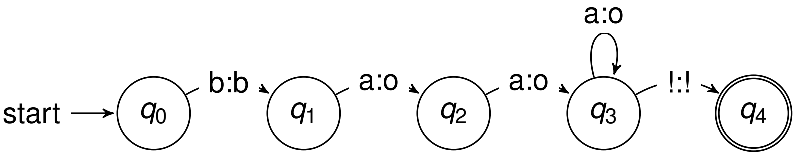
```
baa!

boo!
```



Formal Definition of an FST

To define a FST formally we need to tweak the definition of an FSA to include two more pieces of information.



OUTPUT ALPHABET Δ Now rather than a single alphabet we need two alphabets: the *input alphabet*; and *output alphabet*.

OUTPUT FUNCTION $\sigma(q, i)$ The output function is a mathematical function that takes two arguments (the current state q and a member of the input alphabet i) and returns the associated output characters $o \in \Delta$.

Formal Definition of an FST

Our sheep to ghost language converter example is then formally defined as follow:

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{b, a, !\}$$
$$\Delta = \{b, o, !\}$$
$$q_0 = q_0$$
$$F = \{q_4\}$$

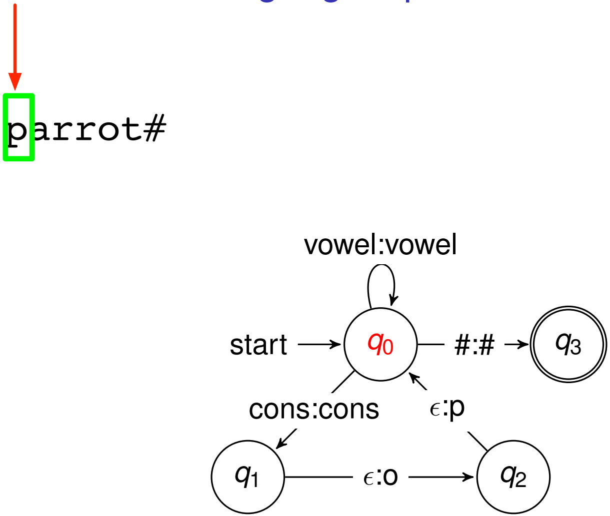
$\delta(q, i) =$

	b	a	!
q_0	q_1	–	–
q_1	–	q_2	–
q_2	–	q_3	–
q_3	–	q_3	q_4
q_4	–	–	–

$\delta(q, i) =$

	b	a	!
q_0	b	–	–
q_1	–	o	–
q_2	–	o	–
q_3	–	o	!
q_4	–	–	–

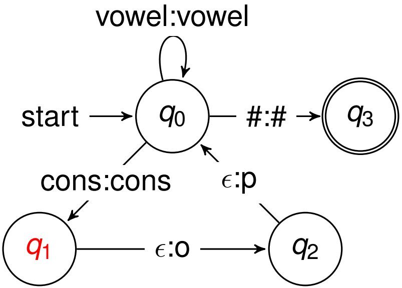
An FST for the language Opish



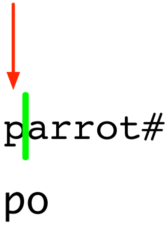
An FST for the language Opish

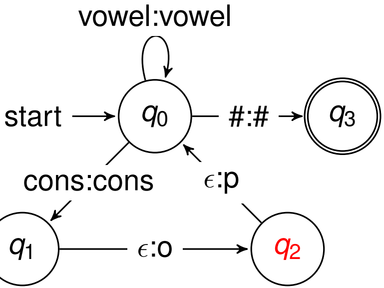
parrot#

p

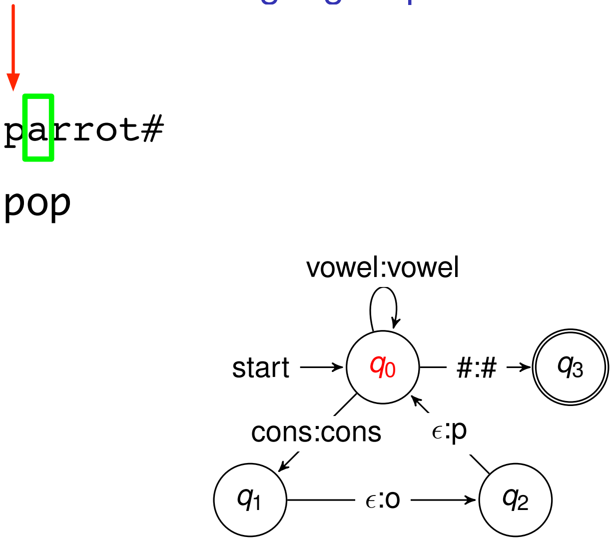


An FST for the language Opish

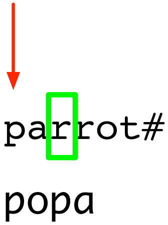


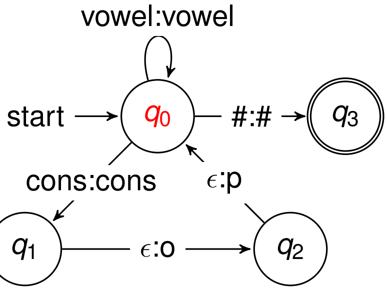


An FST for the language Opish



An FST for the language Opish

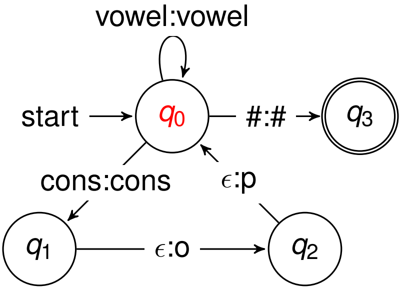




An FST for the language Opish

`parrot#`

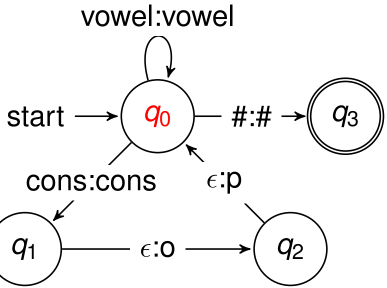
`poparop`



An FST for the language Opish

parrot#

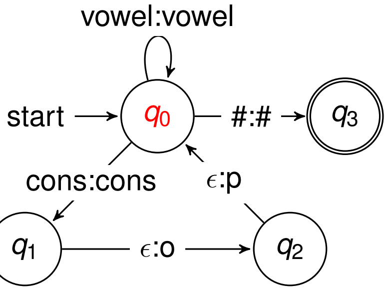
poparoprop



An FST for the language Opish

parrot#

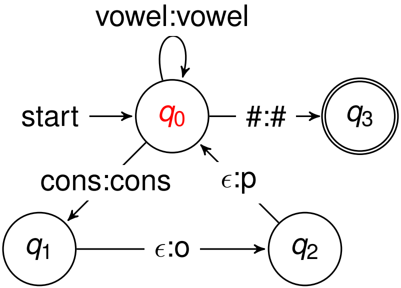
poparopropo



An FST for the language Opish

parrot#

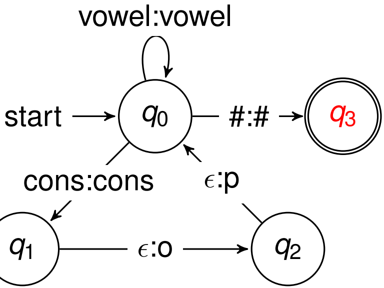
poparopropotop



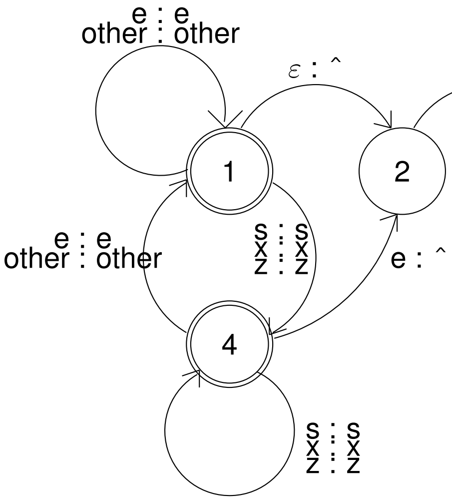
An FST for the language Opish

parrot#

poparopropotop#



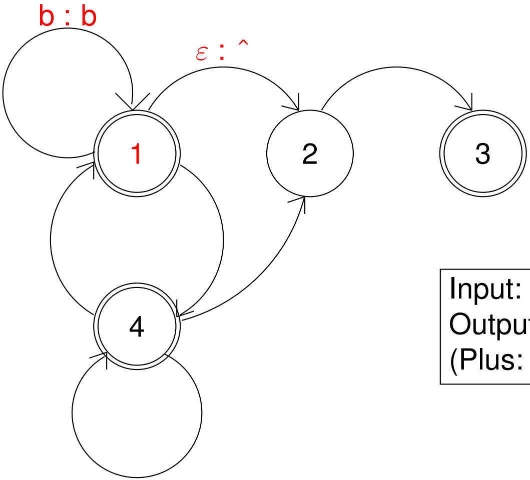
Finite state transducer



$$\varepsilon \rightarrow e / \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} \hat{}\ \_\ s$$

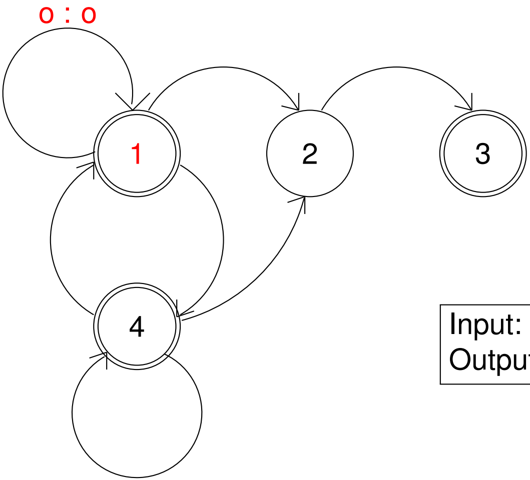
surface : underlying
c a k e s ↔ c a k e ^ s
b o x e s ↔ b o x ^ s

Analysing *b o x e s*



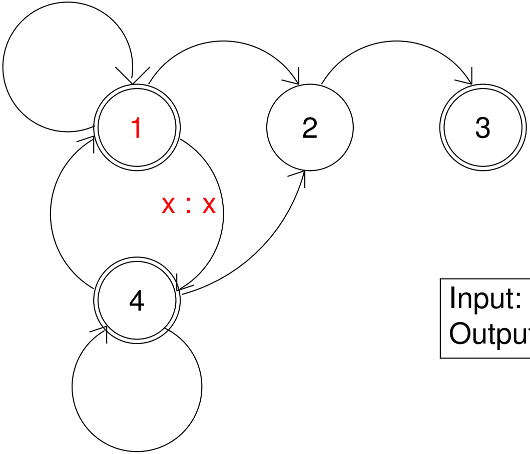
Input: b
Output: b
(Plus: ϵ . ^)

Analysing *b o x e s*



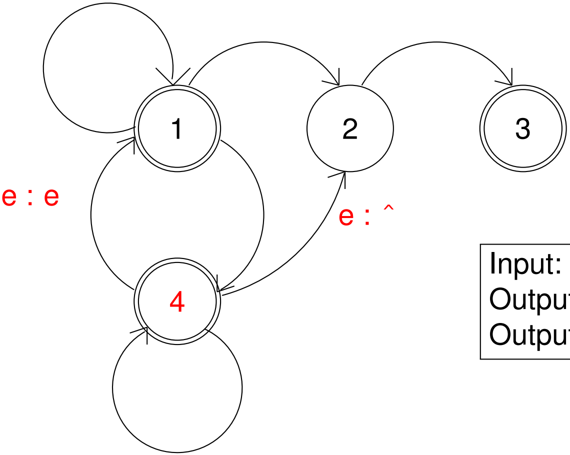
Input: b o
Output: b o

Analysing *b o x e s*



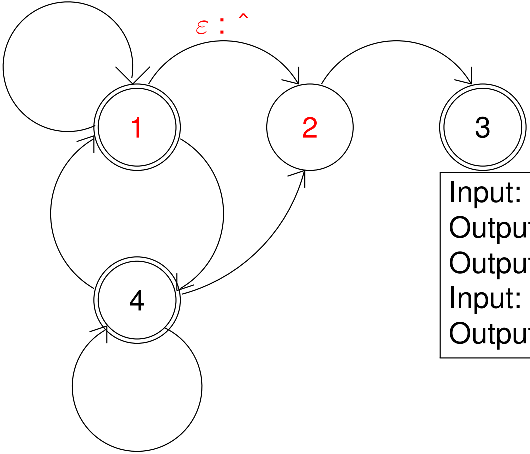
Input: b o x
Output: b o x

Analysing *b o x e s*



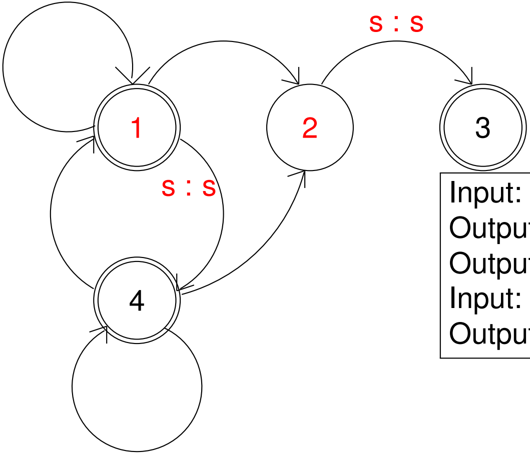
Input: b o x e
Output: b o x ^
Output: b o x e

Analysing *b o x e* ϵ *s*



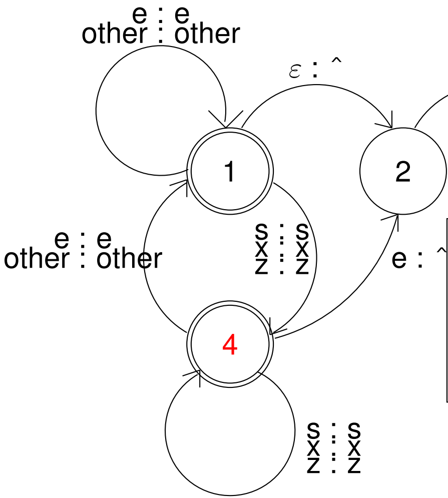
Input: b o x e
Output: b o x ^
Output: b o x e
Input: b o x e ϵ
Output: b o x e ^

Analysing *b o x e s*



Input: b o x e s
Output: b o x ^ s
Output: b o x e s
Input: b o x e ϵ s
Output: b o x e ^ s

Analysing *b o x e s*



Input: b o x e s
Accept output: b o x ^ s
Accept output: b o x e s
Input: b o x e ϵ s
Accept output: b o x e ^ s

Using FSTs

- FSTs assume tokenization (word boundaries) and words split into characters. One character pair per transition!
- Analysis: return character list with affix boundaries, so enabling lexical lookup.
- Generation: input comes from stem and affix lexicons.
- One FST per spelling rule: either compose to big FST or run in parallel.
- FSTs do not allow for internal structure:
  - can't model *un- ion -ize -d* bracketing.
  - can't condition on prior transitions, so potential redundancy

Lexical requirements for morphological processing

- affixes, plus the associated information conveyed by the affix

```
ed PAST_VERB
ed PSP_VERB
s PLURAL_NOUN
```

- irregular forms, with associated information similar to that for affixes

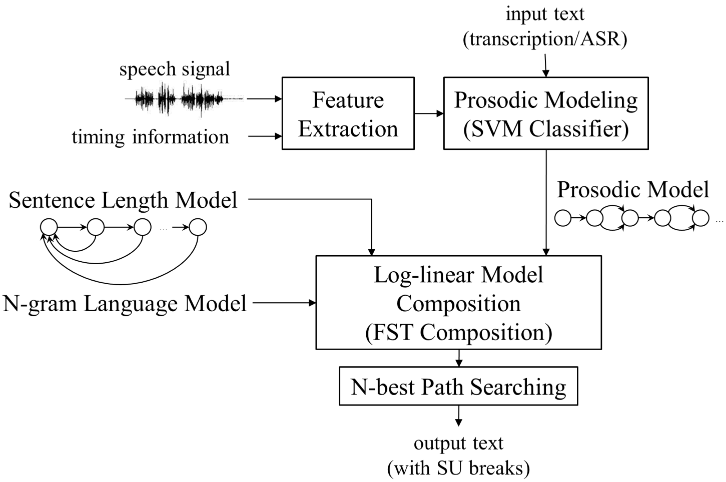
```
began PAST_VERB begin
begun PSP_VERB begin
```

- stems with syntactic categories (plus more)

Some other uses of finite state techniques in NLP

- Grammars for simple spoken dialogue systems (directly written or compiled)
- Partial grammars for text preprocessing, tokenization, named entity recognition etc.
- Dialogue models for spoken dialogue systems (SDS) e.g. obtaining a date:
  1. No information. System prompts for month and day.
  2. Month only is known. System prompts for day.
  3. Day only is known. System prompts for month.
  4. Month and day known.

Lee and Glass sentence segmentation



Concluding comments

- English is an outlier among the world's languages: very limited inflectional morphology.
- English inflectional morphology hasn't been a practical problem for NLP systems for decades.
- Limited need for probabilities, small number of possible morphological analyses for a word.
- Lots of other applications of finite-state techniques: fast, supported by toolkits (eg. openFST), good initial approach for very limited systems.