NLP 201: Morphology and FSTs

Jeffrey Flanigan

University of California Santa Cruz
jmflanig@ucsc.edu

Slides and figures adapted from Jurafsky & Martin, Chris Dyer, Brendan O'Connor, Cagri Coltekin, and Steven Levinson

Fall 2022

Outline

- Morphology background
- Finite state transducers (FSTs)
- Application of FSTs to morphology

Morphology

- The study of words and how they are formed (from other words and morphemes)

Morphemes

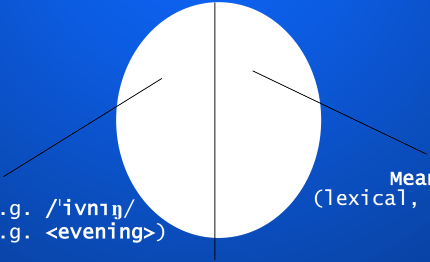
- **A morpheme is a minimal, meaning-bearing unit of language.**
  - Too small (in English): 'p'
  - Too big: 'processing'
- In some languages (Chinese), words and morphemes are basically the same.
- In some languages (Czech, Turkish), most words are made up of several morphemes.
- English is in the middle.

Why morphology?

- Needed for further processing. Want to map words to lemmas (banks to bank, etc)
- Some languages have very complex morphology. The number of words is huge for these languages!
- In order to build applications (Siri, Amazon Alexa), need to analyse these words to their morphemes
- Languages with complex morphology often have very little data. **Have to build the morphological analysers by hand (with FSTs)**.

starting small: revisiting linguistic signs

Morpheme
(minimal pairing of form and meaning)



classification of morphemes

- Morphemes (smallest units that carry meaning)
  - Free morphemes
    - Content words (open class)
      - Nouns
      - Verbs
      - Adjectives
      - Adverbs
    - Function words (closed class)
      - Conjunctions (and, or)
      - Articles (the, a)
      - Demonstratives (this)
      - Prepositions (to, from)
      - Comparatives (more, less)
  - Bound morphemes
    - Affixes
      - prefixes
        - re-
        - un-
        - .
        - .
        - .
      - suffixes
        - -s
        - -ize
        - .
        - .
        - .
    - Bound bases
      - cran-
      - .
      - .
      - .
      - .
    - Contracted forms
      - \`ll
      - \`d
      - \`ve
      - .
      - .

morphemes and their realizations

complex words:	<cats>		<dogs>		<bushes>	
orthographic forms:	<cat>	<s>	<dog>	<s>	<bush>	<es>
phonological forms:	/kæt/	/s/	/dɑg/	/z/	/buʃ/	/ɨz/

plural morpheme {-s}

Allomorphs

/s/ /z/ /ɨz/ (...)

(phonologically conditioned formal variants)

Morphology: Phenomena

- Inflection
- Derivation
- Compounding
- Cliticization

- Inflectional morphology: modify root to a word of the same class, due to grammatical constraints like agreement
- e.g. regular verbs. (Exceptions?)

Morphological Form Classes	Regularly Inflected Verbs			
stem	walk	merge	try	map
-*s* form	walks	merges	tries	maps
-*ing* participle	walking	merging	trying	mapping
Past form or -*ed* participle	walked	merged	tried	mapped

- English is relatively simple

- Derivational morphology: modify root to a word of a different class
  - *derivational*
    derive -ation -al
- Can be tricky
  - *universe --> uni- verse* ?
  - *universal --> uni- verse -al* ???

- Compounding
  - *baseball desktop*

- Cliticization

Full Form	Clitic	Full Form	Clitic
am	'm	have	've
are	're	has	's
is	's	had	'd
will	'll	would	'd

Board Work: Morphology

(5 Min) For the following morphological phenomena, come up with an example in English, and identify the morphemes in the word.

- Inflection
- Derivation
- Compounding
- Cliticization

(3 Min) Discuss with a partner.

Gloss (linguistics)

A series of brief explanations, such definitions and morphological analysis for each word, placed between a sentence and its translation to a different language.
Japanese example:

Naomi-ga	Seiji-o	ut-ta
Naomi-SUBJ	Seiji-O	hit-PAST

'Naomi hit Seiji'

Example: Mandarin Chinese

(12) wǒ men tjɛɛn tsin
I PL play piano
'We are playing the piano' (or 'we are playing the pianos', 'we are going to play the piano', etc.)

(13) ta da wǒ men
s/he hit I PL
'She or he is hitting us', 'she or he will hit us', etc.

Analytic Isolating Language

Example: Chiricahua Apache

hà-ń-ʔàh
out.of-2SUBJ+IMPF-handle.a.round.object+IMPF
'you take a round object (out of enclosed space)'

Polysynthetic Fusional Language

Transparency of word-internal boundaries

- **Isolating language: one-to-one correspondence between morpheme and word**
- **Agglutinating language: a word may consist of several morphemes, but boundaries clearcut**
- **Fusional language: no clear boundary between morphemes, semantically distinct features merged into a single morpheme**

Internal complexity of words

- **Analytic language: one morpheme per word (same as isolating) Actually, more like "close to one morpheme per word." This is on a scale: extreme analytic = isolating**
- **Synthetic language: more than one morpheme per word, but a small number**
- **Polysynthetic language: large number of morphemes per word allowed**

Morphological typology

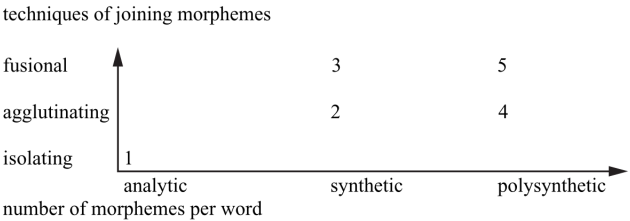


Figure 1.1 Interaction of two types of parameters in word-formation.

Morphological typology

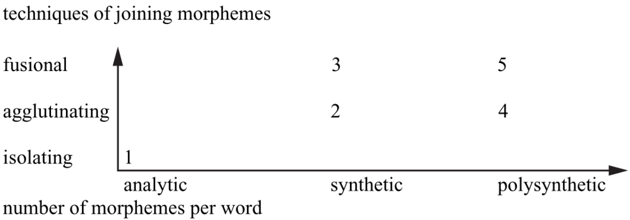


Figure 1.1 Interaction of two types of parameters in word-formation.

Examples

1. Isolating analytic: Vietnamese and Mandarin Chinese
2. Agglutinating synthetic: Hungarian
3. Fusional synthetic: Russian, English
4. Polysynthetic agglutinating: Yupik Eskimo
5. Polysynthetic fusional: Chiricahua Apache

Claim

- Claim: morphology in human languages is **finite state**.
  - Big successes in modeling "morphology" languages like Turkish and Finnish
- Some difficult phenomena
  - Reduplication
  - Circumfixation
  - Root and Template Morphology

Derivational morphology example

- Derivational data we want to model: adjectives become opposites, comparatives, superlatives, adverbs

big, bigger, biggest,	cool, cooler, coolest, coolly
happy, happier, happiest, happily	red, redder, reddest
unhappy, unhappier, unhappiest, unhappily	real, unreal, really
clear, clearer, clearest, clearly, unclear, unclearly	

Derivational morphology example

- Derivational data we want to model: adjectives become opposites, comparatives, superlatives, adverbs

big, bigger, biggest,	cool, cooler, coolest, coolly
happy, happier, happiest, happily	red, redder, reddest
unhappy, unhappier, unhappiest, unhappily	real, unreal, really
clear, clearer, clearest, clearly, unclear, unclearly	

- Task: recognition. Proposed model

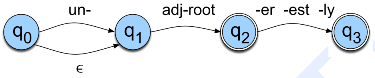


Figure 3.5 An FSA for a fragment of English adjective morphology: Antworth's Proposal #1.

- Any false positives?

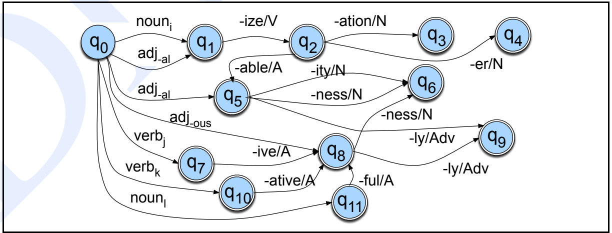


Figure 3.6 An FSA for another fragment of English derivational morphology.

From Recognition to Analysis/Parsing

Full morphological parsing

Ideally, we want to analyse words into their morphemes **(useful for further processing to know tense, number, etc).**

Full morphological parsing

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	'ducks'
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	'duck'
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	'I drink'
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	'I sing'
goose	goose +N +Sg	canto	canto +N +Masc +Sg	'song'
goose	goose +V	puse	poner +V +Perf +1P +Sg	'I was able'
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	'he/she came'
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	'wine'
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	'place'
caught	catch +V +Past			

Figure 3.2 Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

Full morphological parsing

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	'ducks'
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	'duck'
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	'I drink'
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	'I sing'
goose	goose +N +Sg	canto	canto +N +Masc +Sg	'song'
goose	goose +V	puse	poner +V +Perf +1P +Sg	'I was able'
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	'he/she came'
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	'wine'
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	'place'
caught	catch +V +Past			

Figure 3.2 Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

- Need:
  1. Lexicon of stems and affixes **(a lexicon is a list of words)**
  2. **Morpheme ordering model (morphotactics)**
  3. **Spelling changes upon combination (orthographic rules)**
     Ex: city + -s → citys → cities

Important Development

- So far, we've just been using FSAs to represent the *set* of strings in the vocabulary.
- We'd like to go farther, **mapping** strings to deeper analyses: lemma or stem, word type, inflectional features, and so on.
- For this, we need to generalize finite-state automata.

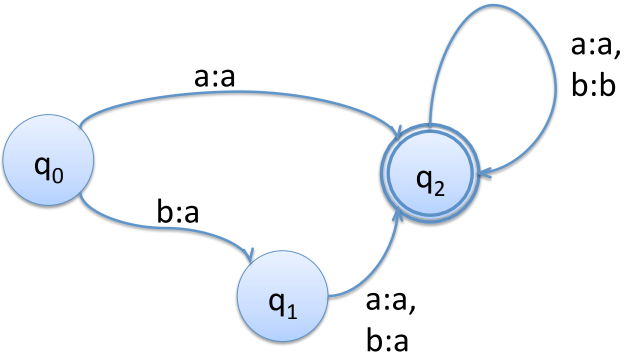
Finite-State Transducers

- Think of an automaton that works with two tapes at the same time.
  - (FSMs only had one tape; we could envision them as acceptors/recognizers, or as generators.)
- The language is a "string-pair" language
- FSTs can be understood as reading or writing either or both tapes!
  - **Recognizer**: take a pair of strings and accept if the pair is in the string-pair language, reject if not.
  - **Generator**: output pairs of strings.
  - **Translator**: Read one string and write out a string. (This is how we will use FSTs for morphological parsing.)
  - **Set relator**: compute relations between sets of strings.

Defining a Finite-State Transducer

Notation	Definition
Q	finite set of states
Σ	finite input vocabulary
Δ	finite output vocabulary
$q_0 \in Q$	start state
$F \subseteq Q$	set of final states
$\delta : Q \times \Sigma^* \rightarrow 2^Q$	transition function; set of possible next states given current state and input sequence
$\sigma : Q \times \Sigma^* \rightarrow 2^{\Delta^*}$	output function; set of possible output strings given current state and input sequence

Example



FSTs and Regular Relations

- "String-pair language" = set of *pairs* of strings.
  - Isomorphic to FSTs in the same way regular languages are isomorphic to FSAs.
- Projection: extract only input or output side.
  - Result of projection is an FSA!
- FSAs are FSTs (identity relation)
- Not closed under difference, complementation, or intersection.
- Closed under: union, inversion (switch input and output labels), **composition**.

Composition

- **composition**: If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from O_1 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2.

- As transducer functions,
  (T1 ∘ T2)(x) = T1(T2(x))

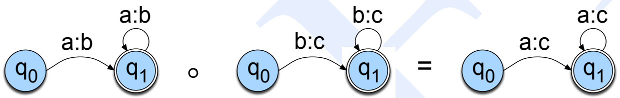


Figure 3.9 The composition of `[a:b]+` with `[b:c]+` to produce `[a:c]+`.

Key Points about Composition

- Composing two FSTs gives us *another FST*
- Because FSAs are a special case of FSTs, we can:
  - Compose an FSA with an FST
    - ("match this input")
  - Compose an FST with an FSA
    - ("match this output")
  - Compose an FSA with an FST and with an FSA
    - ("what are all the ways to get this output from this input?")

Closure properties of FSTs

Like FSA, FSTs are closed under some operations.

- Concatenation
- Kleene star
- ~~Complement~~
- Reversal
- Union
- ~~Intersection~~
- *Inversion*
- *Composition*

Determinism?

- FSTs are nondeterministic in general.
- Not all FSTs can be determinized!
- Sequential FSTs are deterministic on their input.
  - At any state, given each input symbol, there is at most one transition out.
  - Modification: $\delta : Q \times \Sigma \rightarrow Q$ and $\sigma : Q \times \Sigma \rightarrow \Delta^*$
  - Epsilons okay on output, but not the input.
- Generalizations to allow finite amount of ambiguity: *p*-subsequential FSTs.

Non-Determinism

- Actually desirable for NLP!

FSTs for morphological parsing

- A word is understood as a pair of strings: one string is the **lexical level, the other is the surface (spelling as seen in real data).**

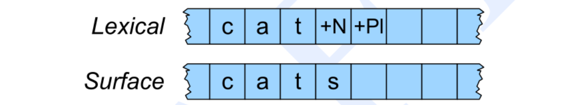


Figure 3.12 Schematic examples of the lexical and surface tapes; the actual transducers will involve intermediate tapes as well.

FSTs for morphological parsing

- A word is understood as a pair of strings: one string is the **lexical level, the other is the surface (spelling as seen in real data).**

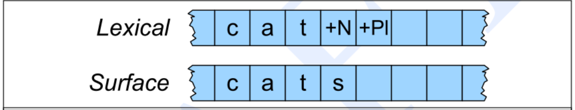
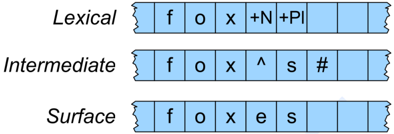


Figure 3.12 Schematic examples of the lexical and surface tapes; the actual transducers will involve intermediate tapes as well.

- The mapping need not be one-to-one!
- 1 lexical string may have many surface strings (**optionality)**
- 1 surface string may have many lexical strings (**ambiguity)**
  No way to resolve the ambiguity. Can be resolved in later processing.

Intermediate string: Two-level morphological analyzers

Often useful to have an intermediate string of bare morphemes before we apply spelling changes (the **orthographic rules)**



ˆdenotes morpheme boundaries

 # denotes end of word

Intermediate string: Two-level morphological analyzers

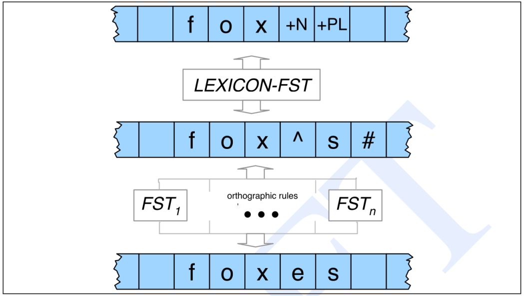


Figure 3.19 Generating or parsing with FST lexicon and rules

Cascades

- Cascading: feeding the output of one transducer in as input to another.
- We can mechanically "fuse" the two transducers together – through **composition** – to get a single transducer that never explicitly represents the intermediate tape.

Crucial thing: FSTs are closed under composition!

Example

Back to our example

	f	o	x	+N	+Pl		
	f	o	x	^	s	#	
	f	o	x	e	s		

lexical transducer is between these two tapes

orthographic transducer is between these two tapes

The lexicon FST

reg-noun	irreg-pl-noun	irreg-sg-noun
fox	g o:e o:e s e	goose
cat	sheep	sheep
aardvark	m o:i u:ϵ s:c e	mouse

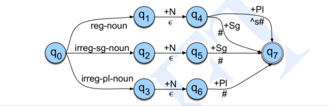


Figure 3.13 A schematic transducer for English nominal number inflection T_{num}. The symbols above each arc represent elements of the morphological parse in the lexical tape; the symbols below each arc represent the surface tape (or the intermediate tape, to be described later), using the morpheme-boundary symbol ^ and word-boundary marker #. The labels on the arcs leaving q_0 are schematic, and need to be expanded by individual words in the lexicon.

Compose

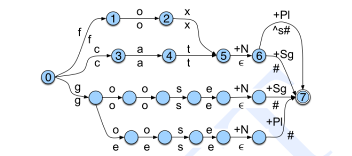


Figure 3.14 A fleshed-out English nominal inflection FST T_{lex}, expanded from T_{num} by replacing the three arcs with individual word stems (only a few sample word stems are shown).

Top FST gives irregular spelling changes, plugged into general FST (middle), to

Orthographic FST

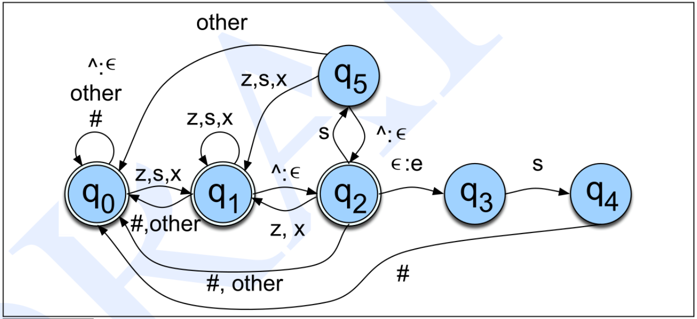


Figure 3.17 The transducer for the E-insertion rule of (3.4), extended from a similar transducer in Antworth (1990). We additionally need to delete the # symbol from the surface string; this can be done either by interpreting the symbol # as the pair #:ϵ, or by postprocessing the output to remove word boundaries.

Example Orthographic Rules

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing*/*-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*,*-z*,*-x*,*-ch*, *-sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

There is an FST for each rule.

Applying Orthographic Rules in Parallel

- More than one orthographic rule might apply to the same word, so we don't want to cascade them.
- Since all rules are constructed to leave strings unchanged that they don't apply to, we can imagine applying the rules in parallel.
- **Intersection** is what we want.
  - But FSTs are not closed under intersection!
  - If strings are always of equal length, we're okay.
  - So treat ε as a standard symbol when intersecting FSTs.

Putting it all together

Compile everything into one big FST

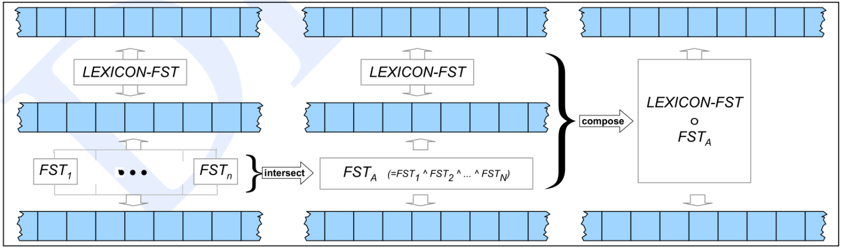


Figure 3.21 Intersection and composition of transducers.

Toward A Parsing Algorithm

- In general, our FSTs will not be deterministic in any sense.
  - Claim: finding the set of valid outputs for a given input is extremely similar to the recognition algorithm for FSAs; just need to do more bookkeeping.
  - As we saw last time, this is a special case of composition.

Other applications of FSTs

- Spelling correction
  Transpositions: *teh* $\longrightarrow$ *the*
- Speech recognition and speech generation (pronunciation dictionaries)
  OpenFST library was originally developed for speech applications
- Some translation models (such as IBM model 1) can be viewed as weighted FSTs
- Transliteration (MT) with weighted FSTs

Remarks

- FSTs can be understood as a flexible, high-level, declarative programming language for working with string relations and sets.
- They can't do everything! But they are a powerful tool for certain kinds of jobs.
- There are nice implementations of FST algorithms, so you can focus on constructing the intuitive modules, then put them together using standard operations.
  - XFST, FOMA, OpenFST, PyFST, Thrax

Further Reading

- Beesley and Karttunen (2003; www.fsmbook.com): a book covering both linguistics and tools for FS morphology.
- Roark and Sproat (2007): the first half covers lots of morphological phenomena and how they can be handled with FSMs.
- Karttunen and Beesley (2005), "Twenty-five years of finite-state morphology," chapter 8 in a festschrift volume for Kimmo Koskenniemi.