Words
Morphology
What’s morphology?

Morphology is the study of the form of words. Words may seem like atomic units of meaning; however, they can often be broken down into smaller atomic units of meaning called morphemes.
What’s a morpheme?

A morpheme is the smallest linguistic unit which has both sound and meaning. There are two types of morpheme:

- Roots/Stems (e.g., <swim>)
- Affixes (e.g., <-ing> <pre->)
A root morpheme can stand by itself. It does not have to be attached to other morphemes although other morphemes can attach to it.
Affixes

Affixes need to be attached to a root morpheme.
There two ways to classify affixes:
  By position
  By use
Affixes by Position

Prefix -- before the root (e.g., <pre->)
Suffix -- after the root (e.g., <-ed>)
Suprafix -- over the root (cf. produce (n) & (v))
Infix -- inside the root (e.g., abso-freaking-lutely)
Affixes by Use

Inflectional
- Does not change word class
- Serves grammatical functions

Derivational
- Changes either word meaning or word class
Attached to nouns:

<-s> as in *John’s book* (Genitive Case)

<-s> as in *Cats are cute* (Plural)
Attached to adjectives:

<-er> as in *warmer* (Comparative)
<-est> as in *warmest* (Superlative)
Attached to verbs:

<-s> as in John swims (3SG-Present)
<-ing> as in John is thinking (Gerund)
<-ed> as in John walked (Past)
<-en> as in John had chosen (Past Perfect)
Excursus: Allomorphs

Plural morpheme <-s> is one morpheme with three phonetic realizations.

- boot-s -> /but -s/
- bee-s -> /bi-z/
- bush-es -> /buʃ -əz/
Derivational Morpheme

Snoozeville
Builder
Predetermination
Relive
Inevitable
Field Exercise
PROLOG BREAK
Five Components of FSA

1. A finite set of states (e.g., \{q0, q2, q3\})
2. A finite input set of symbols (e.g., \{a, b\})
3. A start state \(q0\)
4. A set of final states (e.g., \{q2\})
5. A set of transitions (e.g., \{\delta(q0,b,q1),\delta(q1,a,q2),\delta(q2,a,q2)\})
FSA: Graphically

This FSA accepts the following strings:
ba, baa, baaa, baaaaa, baaaaaaaa, baaaaaaaaa, etc.
FSA: Graphically: Example

- Try doing an example to see which strings get accepted by the FSA
- Try building your own FSA that accepts the strings that meet the requirements
- If you have time, try building the larger language-based FSA
FSA (in Prolog)

q0([b|L]) :- q1(L).
q1([a|L]) :- q2(L).
q2([a|L]) :- q2(L).
q2([]).

---

> q0([baaaa]).
true
Finite State Transducer

An FSA with labeled output
FST Graphically

start \rightarrow q_0 \quad \text{reg-noun: N} \quad q_1 \quad q_2 \quad q_3 \quad q_4

\begin{align*}
q_0 &\quad \text{plural-s: PL} \\
q_1 &\quad \epsilon: \text{SG} \\
q_2 &\quad \epsilon: \text{SG} \\
q_3 &\quad \epsilon: \text{PL} \\
q_4 &
\end{align*}

\begin{align*}
\text{irr-sg-noun: N} \\
\text{irr-pl-noun: N}
\end{align*}
FST (in Prolog)

q0([X|L1],[n|L2]):- reg_noun(X), q1(L1,L2).
q1([X|L1], [pl|L2]):- plural_s(X), q4(L1,L2).
q1(L1, [sg|L2]):- q4(L1,L2).
q0([X|L1],[n|L2]):- irr_sg_noun(X), q2(L1,L2).
q2(L1,[sg|L2]):- q4(L1,L2).
q0([X|L1],[n|L2]):- irr_pl_noun(X), q3(L1,L2).
q3(L1,[pl|L2]):- q4(L1,L2).
q4([],[]).