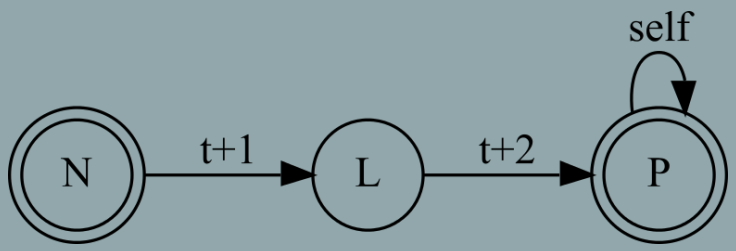
LING-362

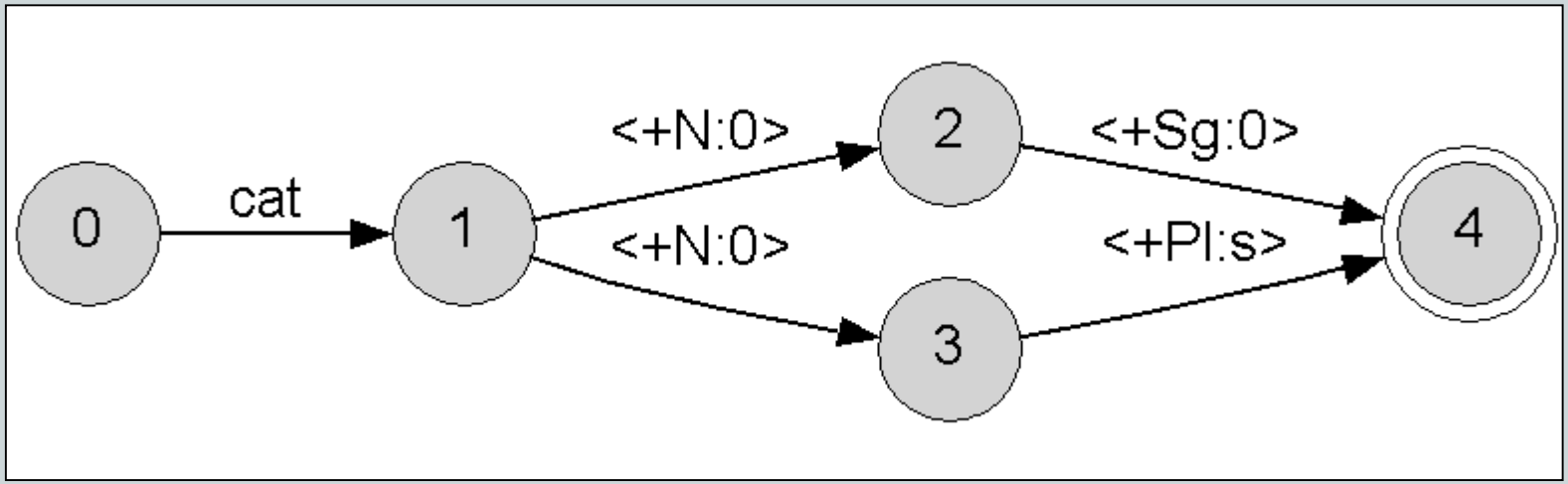
Introduction to Natural Language Processing

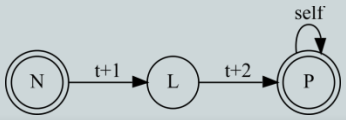
Finite State Methods – review



Transducers as analyzers

- FSMs can Translate a word into an analysis and back:





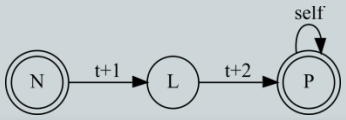
A complex lexicon for English

LEXICON Noun

cat Ninf;
city Ninf;
fox Ninf;
panic Ninf;
try Ninf;
watch Ninf;

LEXICON Verb

beg Vinf;
fox Vinf;
make Vinf;
panic Vinf;
try Vinf;
watch Vinf;



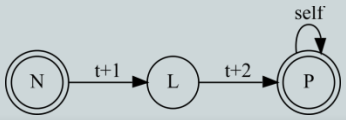
A complex lexicon for English

LEXICON Ninf

+N+Sg:0 #;
+N+Pl:^s #;

LEXICON Vinf

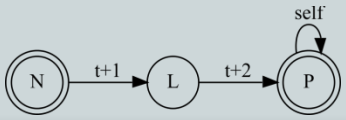
+V:0 #;
+V+3P+Sg:^s #;
+V+Past:^ed #;
+V+PastPart:^ed #;
+V+PresPart:^ing #;



Generating words on either tape

```
transducer = generate_table(options.lexc)
fst = FST(transducer)

print(fst.lower_words(n=3))
```

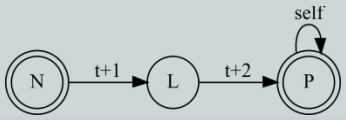
try+N+Pl
try+N+Pl
fox+N+Sg



Generating words on either tape

```
transducer = generate_table(options.lexc)
fst = FST(transducer)

print(fst.upper_words(n=3))
```

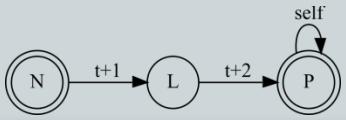
watch^s
try^ed
make^ing



Replacement rules

- We can use re.sub to clean up our outputs in a separate function:

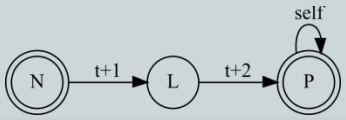
```
def clean_word(word):

    # e-deletion: make^ing -> mak^ing
    cleaned = re.sub(r'e\^(ed|ing)', r'^\1', word)

    # e-insertion: watch^s -> watche^s
    cleaned = re.sub(r'([szx]|ch|sh)\^s', r'\1^s', cleaned)

    # Remove remaining "^"
    cleaned = re.sub(r'\^', '', cleaned)

    return cleaned
```



Combining everything

```
# generate clean words from analyses
print("\nGenerating clean word forms:\n" + "="*20)
to_generate = open(options.inputfile, encoding="utf-8").read()
to_generate = to_generate.strip().split("\n")
for analysis in to_generate:
        generated = fst.transduce(analysis,with_input=False)
        generated = clean_word(generated)
        print(generated)
```

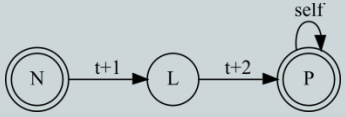
cats
watches
making



Epsilon insertion

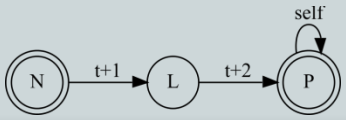
- Note that analysis doesn't quite work, since we expect inputs like "cat^s"
- C++ FSMs can consider producing such symbols from empty input, also called 'epsilon'
- For our pure Python code we can do this:

```
analyzed = fst.transduce(word,with_input=False)

# re.sub cannot invert caret deletion (epsilon insertion)
if analyzed=="" and re.search(r'(s|ed|ing)$',word) is not None:
        with_caret = re.sub(r'(s|ed|ing)$',r'^\1',word)
        analyzed = fst.transduce(with_caret)
```



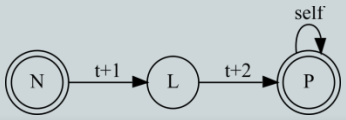
FSM: Going further

- More on XFST syntax: in **Canvas**
- XTAG English Morphology
  - Upenn project for a large coverage English grammar (in TAG, backed by FSM)
  - http://www.cis.upenn.edu/~xtag/swrelease.html
- EMOR (and SMOR for German):
  - http://www.cis.uni-muenchen.de/~schmid/tools/SFST/
- PCKIMMO – English FSM (and Japanese, Finnish)
  - http://www.sil.org/pckimmo
- morpha/morphg – English grammar
  - http://users.sussex.ac.uk/~johnca/morph.html
  - Version ported to Java: https://github.com/knowitall/morpha



Practicing this

- The best way to learn how to build a FSM…
  - Take some language you don't know too well
  - Choose some example words to analyze
  - Compile a grammar that works!
- Today we will do this in a 'hackathon' format
  - Collaborative work
  - Get help, discuss and coordinate
  - The surprise language will be…

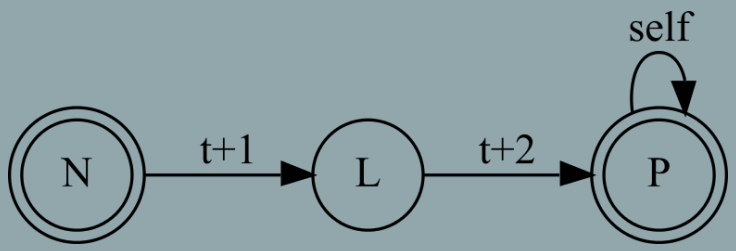


LING-362

Introduction to **N**atural **L**anguage **P**rocessing

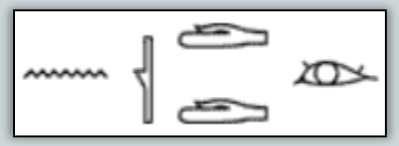
ⲡⲙⲁ ⲛ̄ⲧⲙ̄ⲛ̄ⲧⲥⲉⲁⲓ ⲛ̄ⲧⲙⲟⲣⲫⲟⲗⲟⲅⲓⲁ ⲛ̄ⲧⲙ̄ⲛ̄ⲧⲣⲙ̄̀ⲛ̄ⲕⲏⲙⲉ

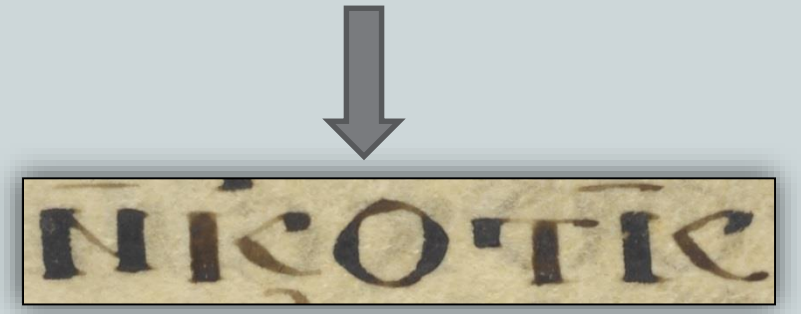
Coptic Morphology Workshop

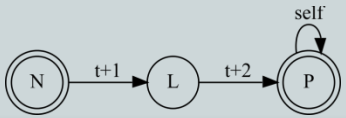


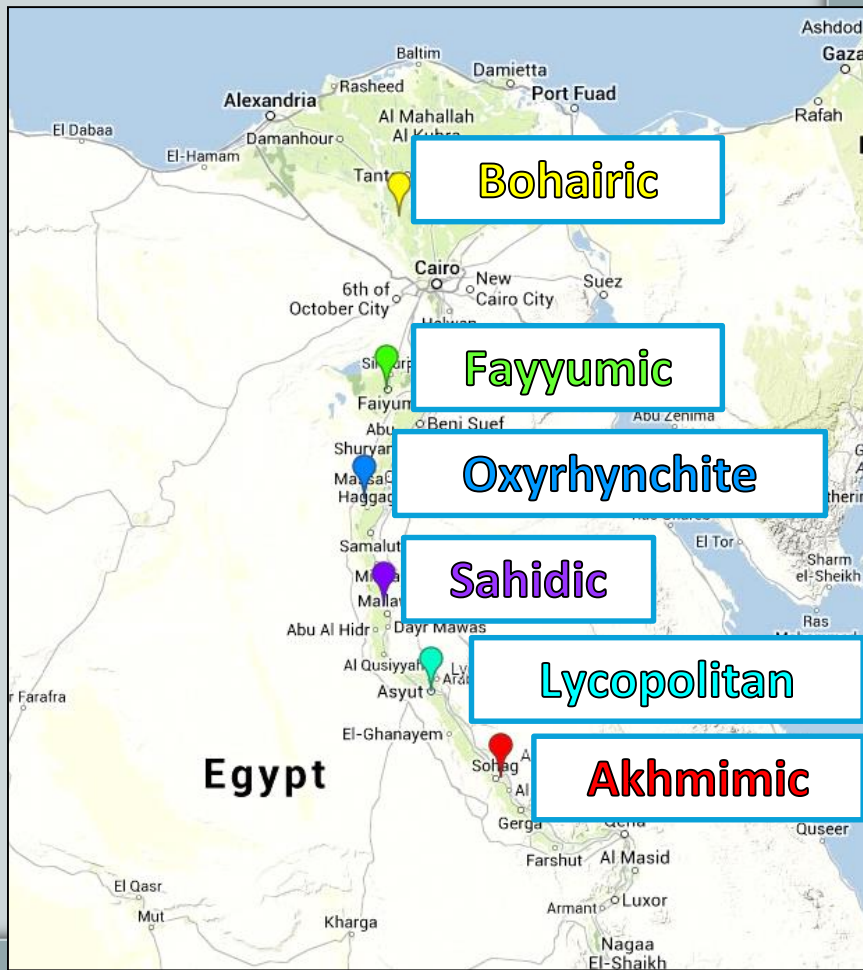
Coptic?

- Last stage of Ancient Egyptian Language (starting 2nd Century)
  - Hellenistic period (1st millennium)
  - Longest continuous documentation
  - Strong contact with Greek
  - Written in Greek letters:







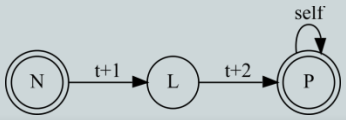


Sample text (simplified)

- a-u-joo-s etbe-t-makaria sara t-parthenos
  PAST-they-say-her about-the-blessed Sarah the-virgin

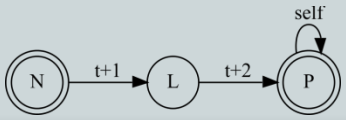
- je-a-s-r-se n-rompe e-s-uHh m-pe-tpe m-p-iero
  that-PAST-she-do-60 of-year while-she-dwell of-the-top of-the-river

- mpe-s-ke-rat-s ebol eneh e-nau e-p-iero
  NGPAST-she-set-foot-her outward ever to-see to-the-river



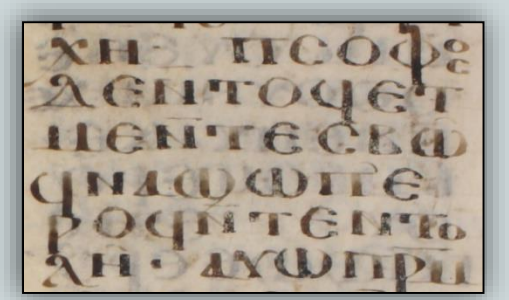
Coptic morphology – the facts

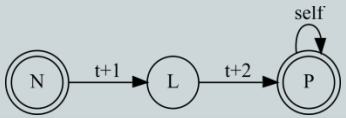
- Nominal bound groups
  - Possible prepositions:
    - ***n*** ‘of’, ***etbe*** ‘about’, ***hn*** ‘in’, ***e*** ‘to’, ***na*** ‘for’
  - Usually an article:
    - ***p*** (masc), ***t*** (fem), ***n*** (plural)
  - Always a noun:
    - ***shime*** ‘woman’
    - ***rOme*** ‘man’
    - ***diabolos*** ‘devil’
    - ***Hi*** ‘house’
    - ***ma*** ‘place’
    - ***rat*** ‘foot’
  - Suffix possessive possible: ***-f*** ‘his’, ***-s*** ‘her’ (then no article)
    - ***rat-s*** ‘her foot’



Coptic morphology – the facts

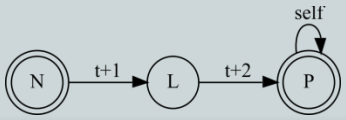
- Verbal bound groups
  - Possible conjugation base:
    - ***a*** ‘PAST’, ***mpe*** ‘NEG-PAST’, ***Sa*** ‘AOR’, ***me*** ‘NEG-AOR’, ***n*** ‘CONJ’
  - Always a subject:
    - Pronoun: ***i*** (1), ***k*** (2), ***f*** (3m), ***s*** (3f), ***n*** (1pl), ***tetn*** (2pl), ***u*** (3pl)
    - If no conjugation base: ***ti*** (1), ***se*** (3pl)
  - Always a verb:
    - ***sOtp*** ‘choose’
    - ***kOt*** ‘build’
    - ***mooSe*** ‘walk’
    - ***jO*** ‘say’ (before object: ***joo***)
  - Possibly an object nominal group
    - Or pronoun: 1sg object is ***t***, 2pl is ***tHutn***, otherwise same as subject with base – *a-f-sotp-t* “he chose me”

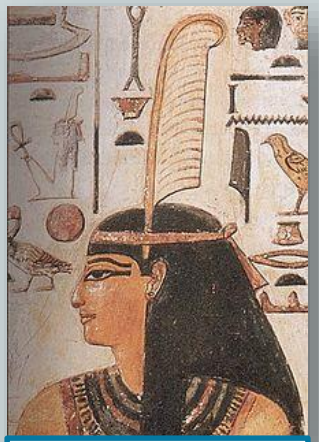




Coptic – ponological rules

- Assimilations:
  - In articles and prepositions, **n** -> **m** before labial (b, m, p): h**n** + p-Hi = h**m**-p-Hi ‘in the house’
  - **k** becomes **g** after n: n+**k**+sOtm -> n**g**sOtm ‘and you hear’
- Epenthesis:
  - **m+t** -> **mnt**: uOm + t = uOm**n**t ‘eat me’
- Verbs are shortened before objects:
  - a-f-s**O**tp – he chose
  - a-f-s**o**tp-f – he chose him
  - a-f-s**e**tp-p-rOme – he chose the man
- Articles are lengthened before clusters:
  - t-me – the truth
  - t**e**-shime – the woman (note: sh = [s+h])

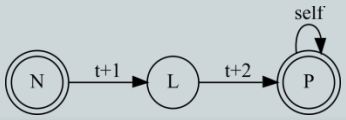




Ma‘at ‘Truth’

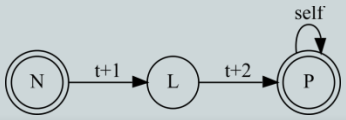
Some tips

- Lexicon contains analysis:form pairs:
  - +N+Pl:^s #;
- For Coptic we can use **glosses** as analyses:
  - **LEXICON** NounM
    man:rOme **#**;
    …
  - **LEXICON** ConjugationBase
    PST+:a SubjPron;



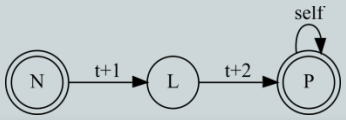
Links

- Sign up for what you're doing right now here:
  - https://corpling.uis.georgetown.edu/ethercalc/coptic
- We will build the lexicon file here:
  - https://corpling.uis.georgetown.edu/etherpad/p/coptic.lexc
- Phonological rules here:
  - https://corpling.uis.georgetown.edu/etherpad/p/coptic.py
- Test cases here: (we want to catch all of them!)
  - https://corpling.uis.georgetown.edu/ethercalc/coptic_tests



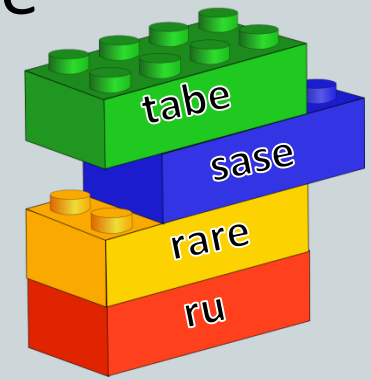
Home work – Japanese verbs

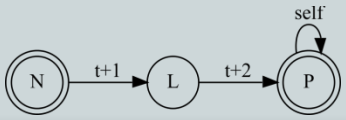
- For ~~**Wednesday**~~ **Friday** we will write a **.lexc** file and **a python script** for Japanese verb forms
- We will practice on four verbs from the two major conjugation classes:
  - -eru/-iru verbs: *taberu* 'eat', *nobiru* 'stretch'
  - -u verbs: *yomu* 'read', *hanasu* 'speak'



Home work – Japanese verbs

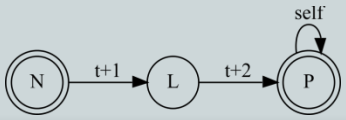
- We will model the causative and passive inflections:
  - -iru/-eru verbs:
    - Drop 'ru'
    - Add **saseru** (causative) or **rareru** (passive)
    - or both: **saserareru** (be made to do something)
    - ***tabesaseru***: make someone eat; ***nobirareru***: be stretched
  - -u verbs:
    - Drop 'u'
    - Add **aseru** (causative) or **areru** (passive)
    - or both: **aserareru**
    - ***yomaserareru***: *be made to read*





Home work – Japanese verbs

- Produce a .lexc file that:
  - Defines the verb stems in each class (you will need two paths of verbal endings)
  - Defines the necessary suffixes (which differ in each class)
  - Combines the suffixes correctly with each verb type
- In a separate **python** script, use the fst's **transduce** command to get analyses for the provided Japanese words file (submit both **.lexc** and **.py** files!)
- You should get all 1+3 possible inflected forms for all 4 verbs (16 forms):
  - ***taberu, tabesaseru, taberareru, tabesaserareru*** (be made to eat)
  - …



Home work – Japanese verbs

- Bonus: [1pt each, total 2pts]
  - Add a gloss to each word in the .lexc file so your analysis also outputs a **translation:**
    - yomaseru -> read+V+Caus
  - Add the honorific suffix -masu to the base form according to this list, and the test forms to the .txt file:
    - Type 1: (use a symbol +Hon)
      - yomimasu
      - hanashimasu <- note: **si** is pronounced **shi** in Japanese - use re.sub!
    - Type 2: (use a symbol +Hon)
      - tabemasu
      - nobimasu