Lecture 27:
Formal Language Theory (2)

Ling 1330/2330 Intro to Computational Linguistics

Na-Rae Han, 12/7/2023

Overview

- HW10: What did you think?

- Formal language theory
  - Eisenstein (2019) Ch.9 Formal language theory, draft copy
  - *Mathematical Methods in Linguistics* by B. Partee, A. ter Meulen and R. Wall
    - Excerpt posted on Canvas, under "Modules"

- Course wrap
- ... and lots of announcements

Are FSA good enough?

Question:

- Is the Finite-State Machine *powerful enough* to capture the grammatical system of English phonology?
- How about English morpho-syntax?
- How about English syntax?

⇐This inquiry forms the basis of the **formal language theory**.

Are all languages *equally complex*?

- Languages over A = {a, b}:
  - L_1 = {x | x is 2 characters long or shorter}
  - L_2 = {x | x contains any number of a's followed by a single b}
  - L_3 = {x | x contains an even number of a's}
  - L_4 = {x | x has form $a^n b^n$}
  - L_5 = {x | x contains equal numbers of a's and b's in any order}
  - L_6 = {x | x is a palindrome}
  - L_7 = {x | x has form *ww*, i.e., consists of two halves that are identical}
  - L_8 = {x | x contains #-many a's where # is a prime number}

"copy" language

- Questions:
  - Are some languages *more complex* than others?
  - Which languages are on the *same complexity scale level*?

Complexity scale

- Languages over A = {a, b}:

❸	L_1 = {x | x is 2 characters long or shorter}
	L_2 = {x | x contains any number of a's followed by a single b}
	L_3 = {x | x contains an even number of a's}
❷	L_4 = {x | x has form $a^n b^n$}
	L_5 = {x | x contains equal numbers of a's and b's in any order}
	L_6 = {x | x is a palindrome}
❶	L_7 = {x | x has form *ww*, i.e., consists of two halves that are identical}
	L_8 = {x | x contains #-many a's where # is a prime number}

- **Complexity scale**
  - $L_1, L_2, L_3 \quad < \quad L_4, L_5, L_6 \quad < \quad L_7, L_8$
- A higher level of complexity requires a **more powerful computing device (=automaton)**.

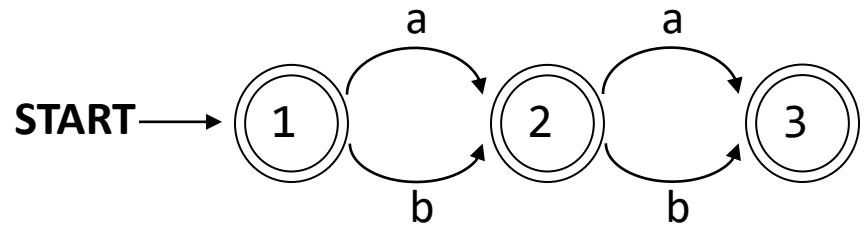
← Which level can be captured by FSA?

Languages definable by a FSA/regex

❸ A language definable by a FSA/regex is called a *regular language*.

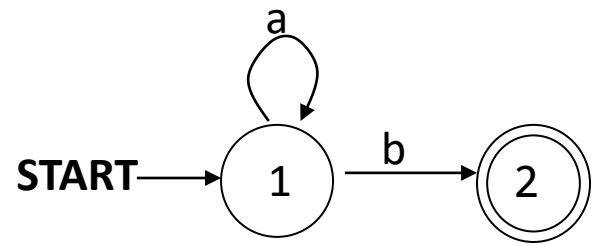
L_1 = {x | x is 2 characters long or shorter}

= (a|b)?(a|b)? ← regex



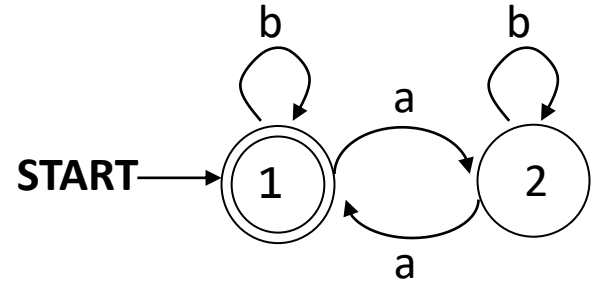
L_2 = {x | x contains any number of a's followed by a single b}

= a*b



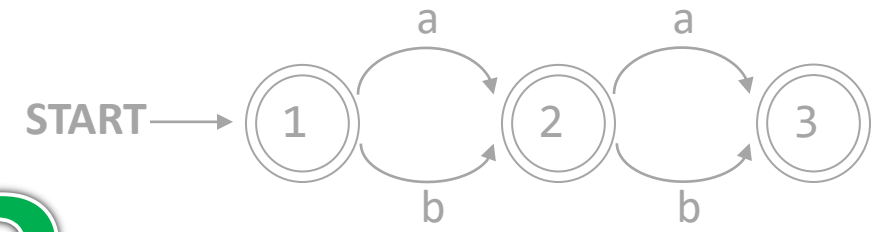
L_3 = {x | x contains an even number of a's}

= b*(ab*ab*)*



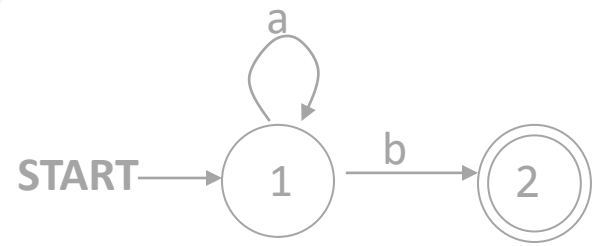
Can these be described by a FSA?

$L_4 = \{x \mid x \text{ has form } a^n b^n\}$



?

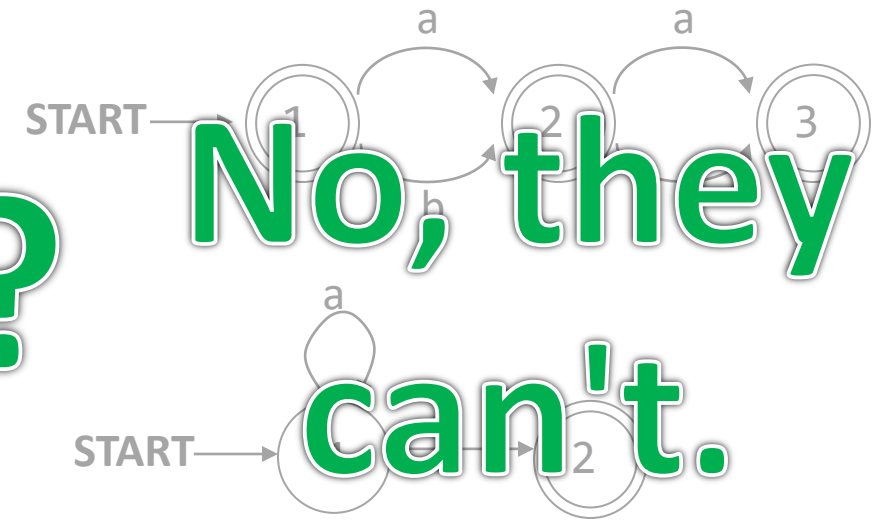
$L_6 = \{x \mid x \text{ is a palindrome}\}$



Can these be described by a FSA?

$L_4 = \{x \mid x \text{ has form } a^n b^n\}$

$L_6 = \{x \mid x \text{ is a palindrome}\}$



They are *not* regular languages.

Complexity scale and automata

- Languages over A = {a, b}:

❸ L_1 = {x | x is 2 characters long or shorter}

L_2 = {x | x contains any number of a's followed by a single b}

L_3 = {x | x contains an even number of a's}

❷ L_4 = {x | x has form $a^n b^n$}

L_5 = {x | x contains equal numbers of a's and b's in any order}

L_6 = {x | x is a palindrome}

❶ L_7 = {x | x has form *ww*, i.e., consists of two halves that are identical}

L_8 = {x | x contains #-many a's where # is a prime number}

Regular languages; can be computed by a **Finite-State Automaton**

Needs a **counting device** (=memory); cannot be computed by a FSA

Complexity scale and automata

- Languages over A = {a, b}:

❸ L_1 = {x | x is 2 characters long or shorter}

L_2 = {x | x contains any number of a's followed by a single b}

L_3 = {x | x contains an even number of a's}

❷ L_4 = {x | x has form $a^n b^n$}

L_5 = {x | x contains equal numbers of a's and b's in any order}

L_6 = {x | x is a palindrome}

❶ L_7 = {x | x has form *ww*, i.e., consists of two halves that are identical}

L_8 = {x | x contains #-many a's where # is a prime number}

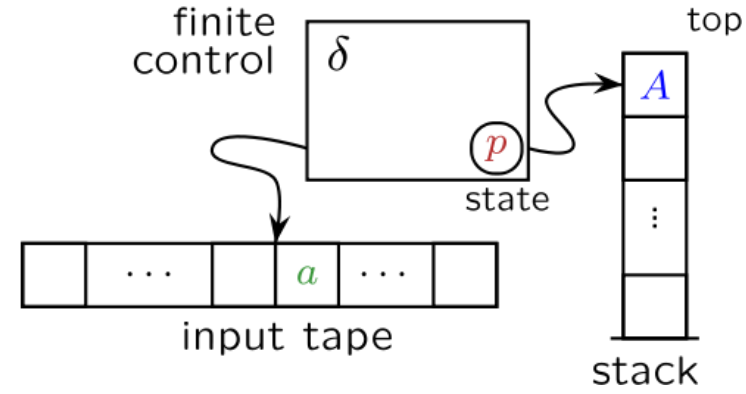
Regular languages; can be computed by a **Finite-State Automaton**

Needs computing machine more powerful than FSA

Needs an *even more powerful* computing machine

Pushdown automata: more powerful

- The pushdown automaton (PDA)
  - Essentially a finite-state automaton with an additional device: an **auxiliary tape where it can read, write, and erase symbols**
  - Tape works like a "**stack**": last-in, first-out
  - Upon reading a symbol in input, it also adds, removes, or exchange the top slot of the stack
  - An input is accepted when:
    - *the entire input has been read, and*
    - *the PDA is in a final state, and*
    - *the stack is empty.*



- There is a PDA that accepts:

  $L_4 = \{x \mid x \text{ has form } a^n b^n\}$

  $L_6 = \{x \mid x \text{ is a palindrome}\}$ (← non-deterministic PDA)

- Languages described by a PDA are called context-free languages.

Complexity scale and automata

❸	L_1 = {x | x is 2 characters long or shorter} L_2 = {x | x contains any number of a's followed by a single b} L_3 = {x | x contains an even number of a's}
❷	L_4 = {x | x has form $a^n b^n$} L_5 = {x | x contains equal numbers of a's and b's in any order} L_6 = {x | x is a palindrome}
❶	L_7 = {x | x has form *ww*, i.e., consists of two halves that are identical} L_8 = {x | x contains #-many a's where # is a prime number}
❶	

Regular languages (finite-state automata)

Context-free languages (pushdown automata)

Context-sensitive languages (linear bounded automata)

More complex languages (Turing machine)

Natural language as formal language

- *Alphabet* (vocabulary) = A = {Bart, Lisa, likes, hates, and, or}
- The largest possible language generated on A:

L_0 = A* = {*e*, 'Bart', 'Lisa', 'Bart Lisa', 'and Bart', 'Lisa Lisa', 'Bart likes Lisa', 'Bart likes Lisa and Lisa likes Lisa', 'or Lisa Bart Bart', ...}

← Any word sequence made out of the vocabulary is grammatical.

← There is no ungrammatical sentence – even '' (=*e*) is well-formed!

← This is an *infinite* set.

- A *language* over A is *any subset* of A*.

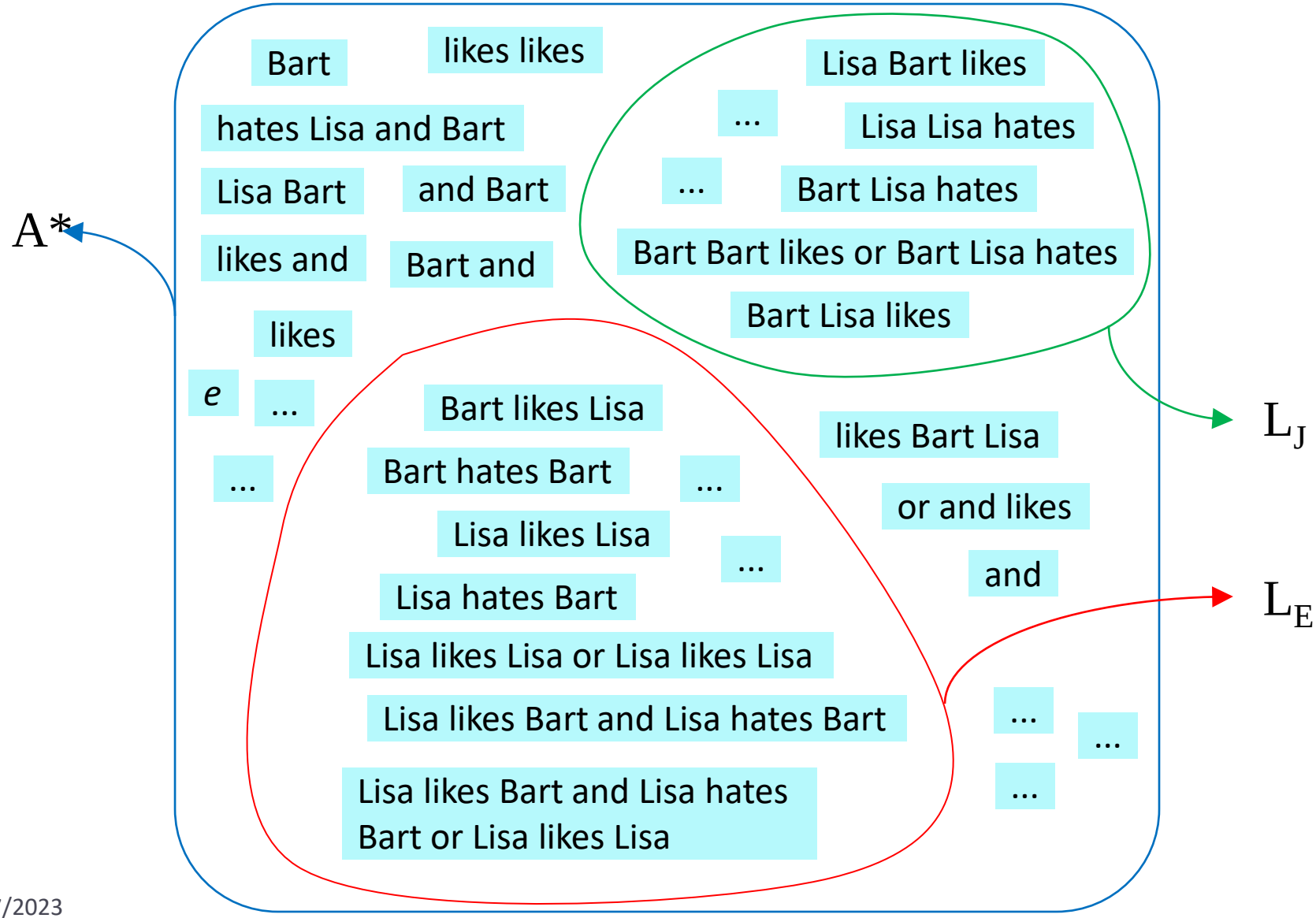
L_E = {'Bart likes Lisa', 'Lisa hates Bart', 'Lisa likes Bart and Bart likes Bart', 'Lisa likes Bart and Bart hates Lisa or Bart hates Lisa', ...}

← L_E is part of English: 'Bart Lisa likes' is ungrammatical for L_E.

L_J = {'Bart Lisa likes', 'Lisa Bart hates', 'Lisa Bart likes and Bart Bart likes', 'Lisa Bart likes and Bart Lisa hates or Bart Lisa hates', ...}

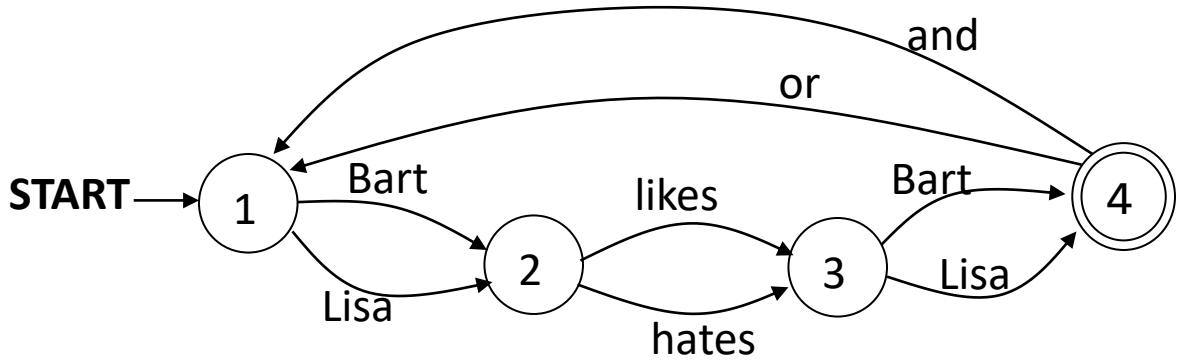
← L_J is Japanese-like: 'Bart Lisa likes' is grammatical.

Languages as sets of strings



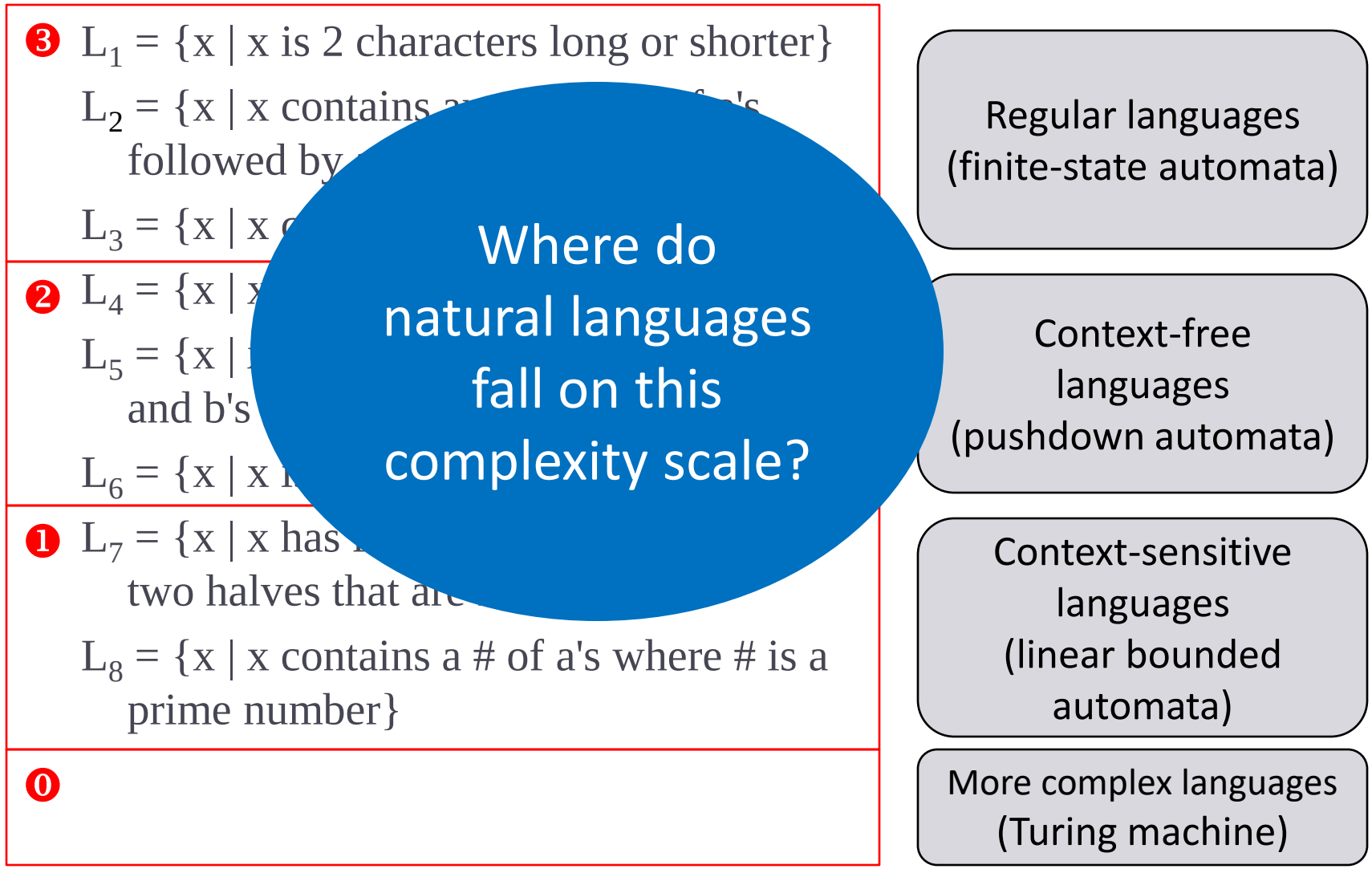
English syntax as FSA

L_E = {'Bart likes Lisa', 'Lisa hates Bart', 'Lisa likes Bart and Bart likes Bart', 'Lisa likes Bart and Bart hates Lisa or Bart hates Lisa', ...}



- So, this toy English language has a FSA representation and therefore is a regular language.
- Questions:
  - Is the ENTIRE English language a regular language?
  - Assuming the language universal, is human language a regular language?

Complexity scale and automata



Natural language syntax: regular or not?

- Is English a regular language?
- Can we find aspects of English syntax that can't be modeled by a FSA?
- How about:
  - *The cat died.*
  - *The cat the dog chased died.*
  - *The cat the dog the rat bit chased died.*
  - *The cat the dog the rat the elephant admired bit chased died.*

- Do you see parallels with:

❷ $L_4 = \{x \mid x \text{ has form } a^n b^n\}$

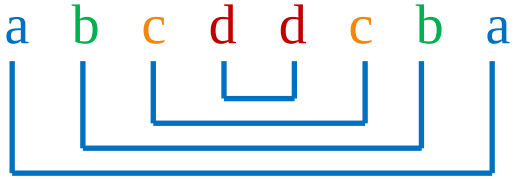
$L_6 = \{x \mid x \text{ is a palindrome}\}$

Context-free languages (pushdown automata)

Nested dependencies

- **Nested dependencies**:
  - *The cat died.*
  - *The cat the dog chased died.*
  - *The cat the dog the rat bit chased died.*
  - *The cat the dog the rat the elephant admired bit chased died.*
- They are a cross between two known context-free languages:
  - Syntactic categories:

    (*the* + common noun)^n Vt^{n-1} Vi
  - Noun-verb agreement:



$L_4 = \{x \mid x \text{ has form } a^n b^n\}$ ❷

$L_6 = \{x \mid x \text{ is a palindrome}\}$

- Mathematically, intersecting two context-free languages results in CFL.

  ← These sentences require at least CFL-level complexity.

  ← **English as a whole is (at least) a context-free language**.

More powerful?

- So, nested dependencies prove that English is not a regular language but a context-free language.
- It means FSA cannot adequately model English; it requires a pushdown automaton.
- By extension, this proves that human language as a whole is at least a context-free language.
- Question:
  - Is context-freeness enough?

  = Can pushdown automata model *all* aspects of human language?

  = Are there any aspects that require an *even more powerful* computing machine?

Beyond context-free

- **Cross-serial dependency** in Swiss German:
  - Jan säit das mer em Hans es huuns hälfed aastriiche

    John said that we Hans$_{\text{-Dat}}$ the house$_{\text{-Acc}}$ helped paint

    "John said that we helped Hans paint the house."
  - Jan säit das mer d'chind em Hans es huuns lönd hälfed aastriiche

    John said that we the-kids$_{\text{-Acc}}$ Hans$_{\text{-Dat}}$ the house$_{\text{-Acc}}$ let help paint

    "John said that we let the children help Hans paint the house."

- Can these sentences be modeled by a pushdown automaton?
  - No. This construction is analogous to:

❶ $L_7 = \{x \mid x \text{ has form } ww, \text{ i.e., consists of two halves that are identical ("copy language")}\}$

Context-sensitive languages (linear bounded automata)

Human language is context-sensitive

- **Cross-serial dependency** in Swiss German:
  - Jan säit das mer d'chind em Hans es huuns lönd hälfed aastriiche

John said that we the-kids$_{-Acc}$ Hans$_{-Dat}$ the house$_{-Acc}$ let help paint

"John said that we let the children help Hans paint the house."

- Cross-serial dependencies require something more powerful than a pushdown automaton.
  - Swiss German is more complex than context-free languages.
  - **Human language as a whole is not context-free; it is context-sensitive in terms of complexity scale.**

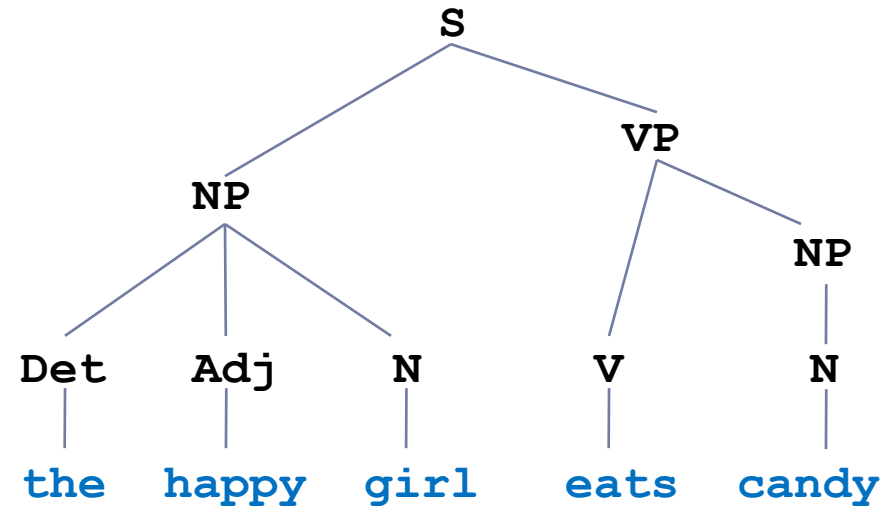
← Turns out, there are finer levels within context-sensitiveness;

← Human language can be shown to be only **mildly context-sensitive**.

But what about trees and rules?

- A ‘tree’ structure for *The happy girl eats candy*:

- Rules used:
  - `S → NP VP`
  - `VP → V NP`
  - `NP → Det Adj N`
  - `NP → N`
  - `Det → the`
  - `Adj → happy`
  - `...`



- Phrase structure rules can also be subjected to formal treatment.

A finite device to describe an infinite set

- **A language is potentially *infinite*.**

  (All *interesting* languages are infinite. The vocabulary is always finite.)

- We need **a *finite* device** that describes all of the grammatical strings in the language to the exclusion of all ungrammatical strings.
- **Computing machines**
  - ex. Finite-state automata, push-down automata, linear bounded automata, Turing machine
  - Functions as a *recognizer*: accepts grammatical strings and rejects ungrammatical strings.
- **Grammar**
  - ex. Phrase-structure grammar, transformational grammar
  - Functions as a *generator*: generates grammatical strings.

A formal definition of grammar

- A **formal grammar** (or simply a grammar)
  - is a deductive system of axioms and rules of inference, which generates the sentences of a language as its theorems.
  - A grammar consists of:
    - V_T (a set of terminal alphabet) = {a, b}
    - V_N (a set of non-terminal alphabet) = {S, A, B}
    - S (the initial symbol : a member of V_N)
    - R (a set of rules) =

      S → ABS
      S → *e*
      AB → BA
      BA → AB
      A → a
      B → b
  - Rules operate as "rewriting rules": starting from the initial symbol, rules are applied to any substring to yield a new string until the string entirely consists of terminal symbols.
  - The *language* generated by a grammar is the set of all strings generated.

In English, please?

- A grammar:
  - V_T (a set of terminal alphabet) = {Mary, sings}
  - V_N (a set of non-terminal alphabet) = {S, NP, VP}
  - S (the initial symbol : a member of V_N)
  - R (a set of rules) =

    S → NP VP S
    S → *e*
    NP VP → VP NP
    VP NP → NP VP
    NP → Mary
    VP → sings

- What do you think of this phrase structure grammar?
  - What do you think of the rules?
- What kind of language does it generate?
- Does English need a grammar like this?
- Does English grammar need **restrictions** on *what **types** of rules* are and are not allowed?

Too powerful

- A grammar:
  - V_T (a set of terminal alphabet) = {Mary, sings}
  - V_N (a set of non-terminal alphabet) = {S, NP, VP}
  - S (the initial symbol : a member of V_N)
  - R (a set of rules) =

    S → NP VP S
    S → *e*
    NP VP → VP NP
    VP NP → NP VP
    NP → Mary
    VP → sings

- This grammar allows many different forms of rewriting rules.
- It generates strings with an equal number of 'Mary' and 'sings', in any order
- We don't need rules like NP VP → VP NP, at least for English
- Turns out, this grammar isn't very restricted → is a form of grammar with the greatest generative power.

Generative power of grammar

- A grammar:
  - V_T (a set of terminal alphabet) = {Mary, sings}
  - V_N (a set of non-terminal alphabet) = {S, NP, VP}
  - S (the initial symbol : a member of V_N)
  - R (a set of rules) =

S → NP VP S	NP → Mary
S → *e*	VP → sings
NP VP → VP NP	
VP NP → NP VP	

- Grammars come with their own **generative power**.
- A grammar can be too powerful → leads to overgeneration
- By placing restrictions on the *form* of the rules, one can restrict what type of string rewriting is possible and therefore restrict the power of the grammar.
- **As linguists, we are interested in finding a form of grammar that is powerful enough for all human languages but is not overly powerful.**

Classes of grammar

- **The Chomsky Hierarchy**
  - By putting increasingly stringent **restrictions** on the allowed forms of rules, we can establish a series of grammars with decreasing generative power.

- **Type 0**: any rules allowed
- **Type 1**: each rule is of the form $\alpha A \beta \rightarrow \alpha \psi \beta$, where $\psi \neq e$
- **Type 2**: each rule is of the form $A \rightarrow \psi$
- **Type 3**: each rule is of the form $A \rightarrow xB$ or $A \rightarrow x$

- α, β, ψ: arbitrary strings (consist of terminal and non-terminal alphabets; can be empty)
- A, B: a non-terminal symbol
- x: a terminal symbol

Example of Type 2 grammar

- Type 2: each rule is of the form A → ψ
  - A grammar containing:
    - V_T (a set of terminal alphabet) = {a, b}
    - V_N (a set of non-terminal alphabet) = {S}
    - S (the initial symbol : a member of V_N)
    - R (a set of rules) = { S → aSa, S → bSb, S → a, S → b, S → *e* }

- What kind of language does it generate?
  - Answer: $L_6 = \{x \mid x \text{ is a palindrome}\}$
- Can the palindrome language be described by a Type 3 grammar? (each rule is of the form A → xB or A → x)
  - Answer: NO.

Classes of grammar

- **The Chomsky Hierarchy**
  - **Type 0**: any rules allowed
    - Called ***unrestricted rewriting systems***
  - **Type 1**: each rule is of the form αAβ → αψβ, where ψ ≠ *e*
    - Lets us specify context: αAβ → αψβ is the same as A → ψ / α__β!
    - Called ***context-sensitive grammar***
    - Languages it describes: ***context-sensitive languages***
  - **Type 2**: each rule is of the form A → ψ
    - Called ***context-free grammar***
    - Languages it describes: ***context-free languages***
  - **Type 3**: each rule is of the form A → xB or A → x
    - Called ***regular grammar***
    - Languages it describes: ***regular languages***

Languages, automata, and grammar

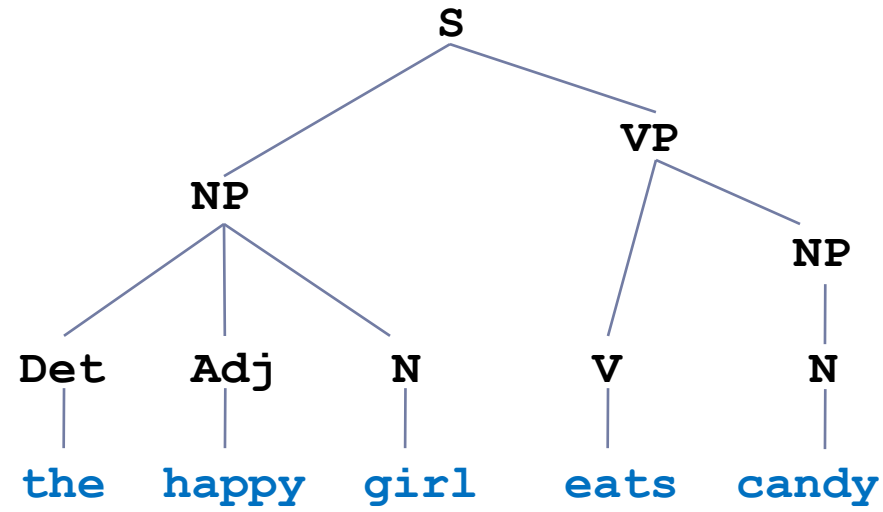
- The Chomsky Hierarchy fits with the complexity scale.

Language example	Language class	Automaton	Grammar
	Type 0 languages	The Turing machine	Type 0 grammar
$L_7 = \{x \mid x \text{ has form } ww\}$ ("copy language")	Context-sensitive languages	Linear bounded automaton	Type 1 grammar (context-sensitive grammar)
$L_6 = \{x \mid x \text{ is a palindrome}\}$	Context-free languages	Pushdown automaton	Type 2 grammar (context-free grammar)
$L_2 = a^*b$	Regular languages	Finite-state automaton	Type 3 grammar (regular grammar)

  - Which grammar is "Phrase structure grammar"?
  - Which grammar formalism is frequently utilized in NLP?

Phrase structure grammar

- Rules used:
  - S → NP VP
  - VP → V NP
  - NP → Det Adj N
  - NP → N
  - Det → the
  - Adj → happy
  - ...

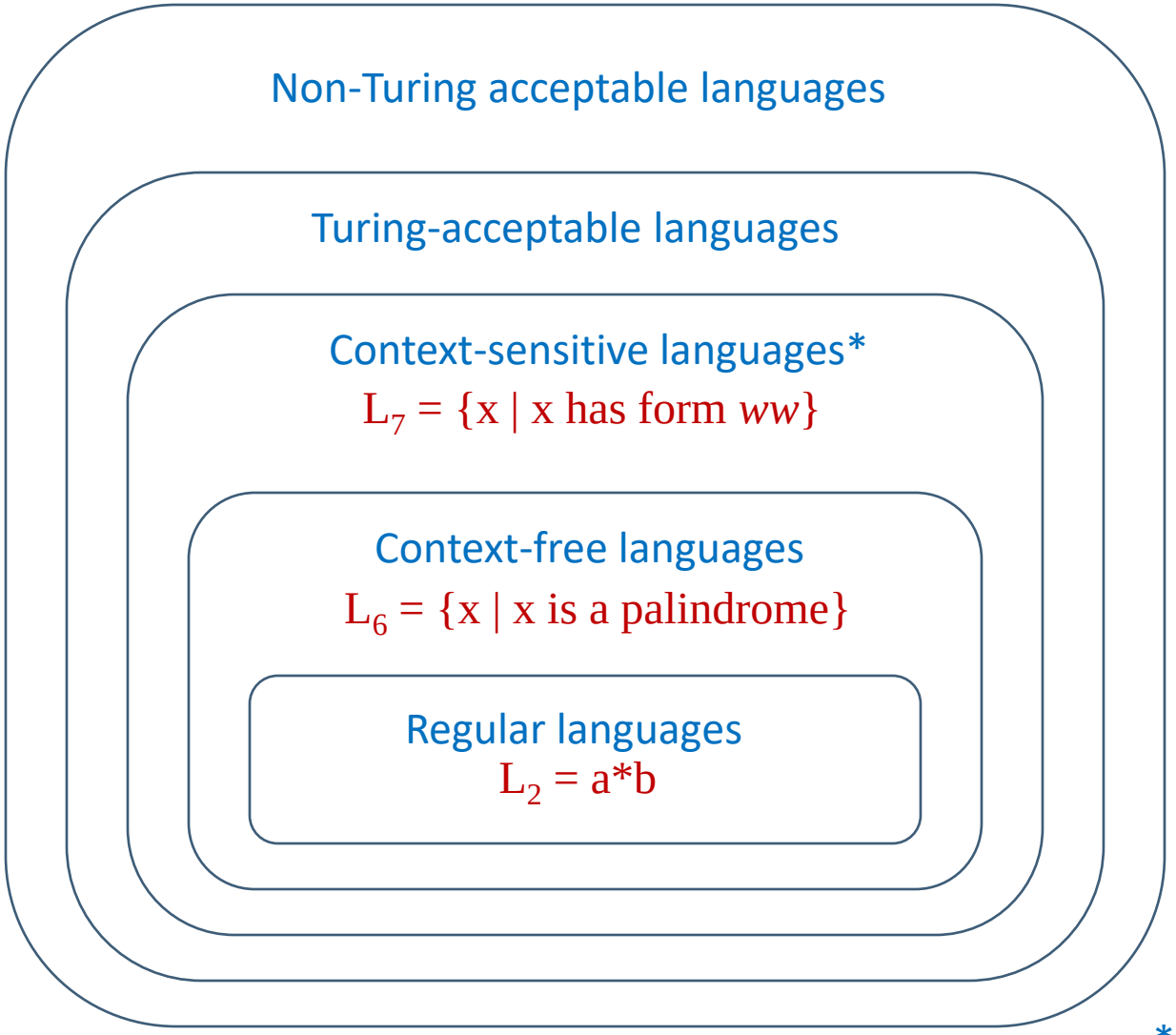


- **Type 0**: any rules allowed
- **Type 1**: each rule is of the form αAβ → αψβ, where ψ ≠ *e*
- **Type 2**: each rule is of the form A → ψ
- **Type 3**: each rule is of the form A → xB or A → x

Context-Free Grammar (CFG)

- α,β,ψ: arbitrary strings (consist of terminal and non-terminal alphabets; can be empty)
- A, B: a non-terminal symbol
- x: a terminal symbol

Inclusion relations in formal languages



- **Inclusion relationship**:

  a regular language is a context-free language, a context-free language is a context-sensitive language, etc.

* excluding {e}

Natural language morphology: regular or not?

- Are there aspects of morphology that cannot be modelled by FSA?
- YES:
  - long-distance dependencies (*un-drink-able* vs. **un-drink*)
  - templatic morphology (Arabic)
- Oh no!! Abandon FST and FOMA!
  - Not so fast.
  - It's true FST in its pure implementation cannot handle the above phenomena…
  - However! Foma and FST-based systems (XFST, etc.) come with additional devices for handling them on a limited/bounded basis:
- **Flag diacritics** in Foma/XFST and **long-distance dependencies**
  - https://fomafst.github.io/morphtut.html#Advanced:_long-distance_dependencies_and_flag_diacritics

Course Wrap Up

You learned this semester:

- Text encoding systems, Unicode
- How spell checkers work
- Corpus linguistics: type, token, TTR, Zipf's law
- Basic text processing and stats: tokenization, frequency distribution, conditional frequency distribution
- *n*-gram language models
- Machine learning and document classification
- Evaluation of machine learning systems
- Naïve Bayes classifier
- Regular expressions and finite-state automata
- Computational morphology: FST
- Part-of-speech (POS) tagging: *n*-gram taggers and HMMs
- Syntactic tree representation, context-free grammar, dependency grammar, parsing
- Computational semantics: WordNet, logic-based, PropBank, vector semantics
- Core concepts in Information Theory: TF-IDF, noisy channel model
- Fundamentals of machine translation (MT) systems: classic, SMT, NMT
- Formal language theory and the Chomsky Hierarchy
- The state-of-the-art of NLP/AI, LLMs, societal impact, future prospect

What we did *not* cover

- Computational phonology (did a little bit with Foma)
- Speech processing & synthesis
- Natural language *generation*
- Question answering and summarization
- Dialogue systems and conversational agents
- Off-the-shelf NLP solutions
- More sophisticated machine learning algorithms:
  - Maximum entropy (ME), conditional random fields (CRF), support vector machine (SVM), deep learning…

Join PyLing!

- **Pitt Python Linguistics Group (PyLing)**
  - On **Discord**! Invitation link on MS Teams.
  - Also: **email list** (let me know)
- Open to LING1330/2330 alums and all linguists/NLP folks who like doing things in Python
- Meet every few weeks over snacks
- Practice Python, chat about computational linguistics, guest speakers, other fun activities

- Studying CL at Pitt: a Guide
  - https://sites.pitt.edu/~naraehan/computational_linguistics.html

Wrapping up

- **Do the OMET survey!**

- Homework 10
  - On MS Teams: Share your HW10 essay, leave comments, gain **extra credit**
  - Upload yours NOW (by Sunday night), and share comments!

- Na-Rae's special office hours on Monday
  - **Monday (12/11) 1-2:45pm**, in person in G17 CL and on MS Teams

- Grades, late work forgiveness ➔
- Extra credit ➔
- Final exam info ➔

Your grade: what's ahead

- **Canvas's Grade Center** is being prepped
  - Your exercise score is in
  - Homework 9 and 10 grades are outstanding
  - Attendance & participation records (will post 2nd half attendance soon)
    - **1 missed class exemption → raised to 2**
  - Weighted running total (CAVEAT!!)

- **Late work forgiveness**
  - Everyone gets one make-up opportunity. Choose from:
    1. Finish up an incomplete homework submission or re-do a part, no penalty.
    2. Up to 3 days of late submission penalty waived.
    3. Missed homework: 25% penalty. Upload on Canvas and email me.
    4. Missed exercise: 5/10 for satisfactory (80+%) work. Email me as attachment.
  - Deadline: **12/15 (Fri) 11:59pm**. Email me and let me know of your choice!
  - If a solution has been published, feel free to look it up. It's fine as long as you don't blindly copy it. (Make sure to demonstrate you are not blindly copying.) There's already a late penalty, and I'd rather you learn.

Extra credit, round-up

- If you have 100% on Exercises, you are already eligible for a standard round-up, up to 0.4%.
  - Normally 89.6% is B+; it will be bumped up to A- (90%)

- Extra credit opportunity (1): NLP talk
  - Attend Linguistics dept colloquium:
    - Dec 1 (Fri) 3pm, G8 CL. Lorraine Li, "Probabilistic (Commonsense) Knowledge in Language"
    - If you can't attend this one, find a different CL/NLP talk (CMU, Pitt, online)
  - Submit a short report on Canvas, earn 0.3% extra credit

Both due 12/15 (Friday) 11:59pm

- Extra credit opportunity (2): share HW10 essays
  - On MS Teams, share your HW 10 essay (by this Sunday), read 3 classmates' essays and leave comments, earn 0.3% extra credit

Final exam

- 12/13 (Wed), 4—5:50pm
- At G17 CL (Language Media Center)

- 150 total points (50% larger than midterm)
- All pen-and-pencil based.
- **1 cheat sheet allowed**:
  - letter-sized, front-and-back, hand-written.
- Cumulative! 10-20% will be from first half of the semester.
- Make sure to study book chapters and other linked materials. Post-midterm, my slides are not as "comprehensive".