Natural Language Processing

Lecture 13: The Chomsky Hierarchy

Formal Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols, N
- Special start symbol $S \in N$
- Production rules
  - Restrictions on the rules determine what kind of grammar you have

A formal grammar G defines a **formal language**, usually denoted L(G).

Regular Grammars

- NT → T NT
- NT → T
- If first symbol after arrow is a Terminal
- Second is a NonTerminal

Regular Grammars

- aaaa...bbbb...
- S → a AS
- S → a BS
- AS → a AS
- AS → a BS
- BS → b
- BS → b BS

Regular Grammars

- L(RG) can be recognized by FSM
- Can be determinized
  - Each state has at most one arc per terminal
  - But might need 2^n new states
- Can be minimized
  - Can find a minimal set of states/arcs that
  - Accept the same language
- Used in regular expressions

Context Free Grammars

- NT → NT NT T
- NT → T NT
- Only one non-terminal on left hand side
- No restriction on right hand side.
- Good for “bracketing”

CFGs

- $S \rightarrow S + S$
- $S \rightarrow S - S$
- $S \rightarrow (S)$
- $S \rightarrow a, b$
- For arithmetic expressions with a,b

Context Free Grammars

- L(G) recognized as push-down automata
- Can be normalized
  - Chomsky normal form
  - Only one or Two symbols on rhs
- Most programming languages are context free languages

Context Sensitive Grammars

- NT (NT) NT → T NT
- NT (NT) _ → T
- Lhs can be more than one symbol
- Bracket symbol rewrites to rhs
- Often used in phonological rules
  - A → B c __ d
  - /n/ → /m/ * __ [/p/ /b/]

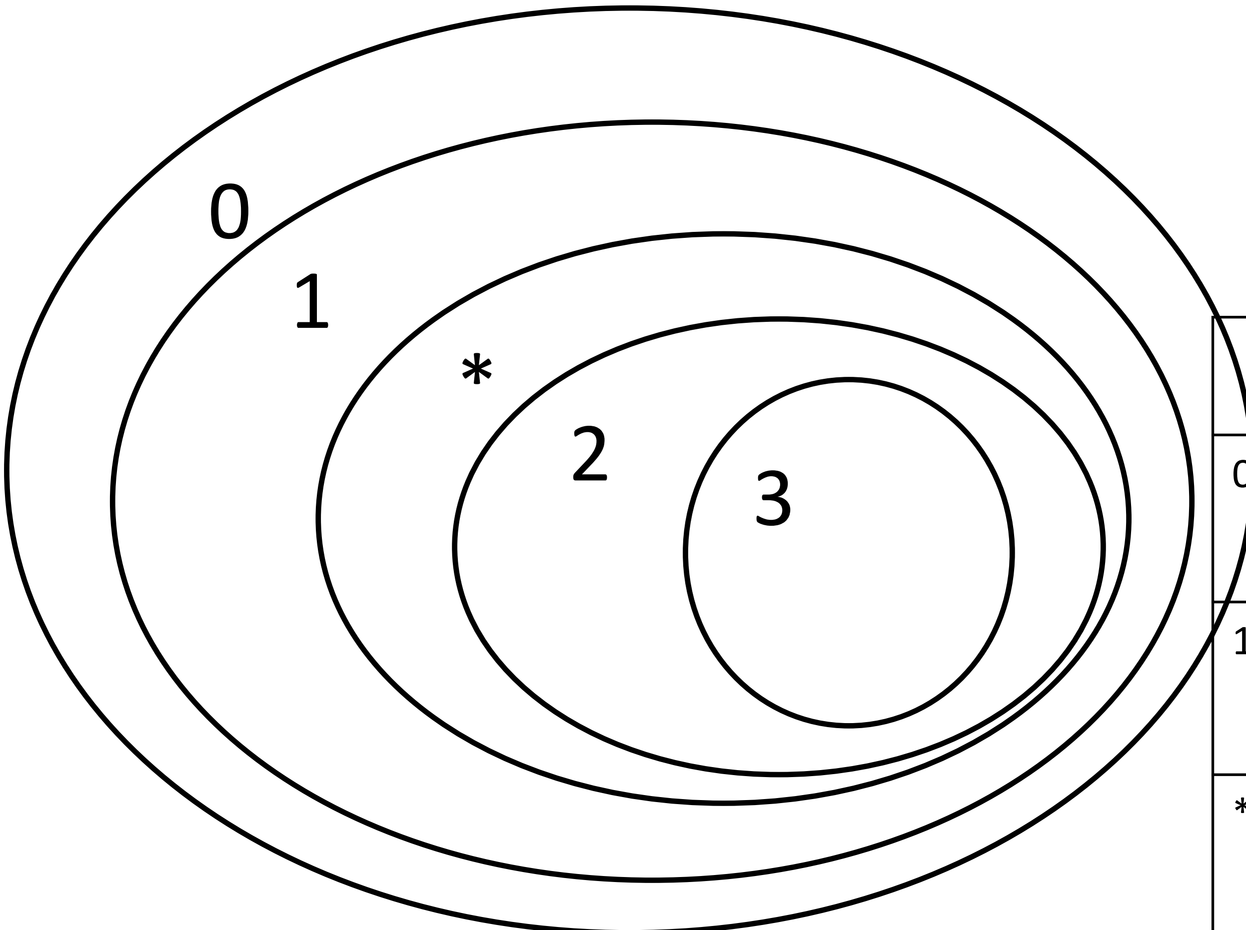
Context Sensitive Grammars

- L(G) recognized by linear bounded automata
- Can be harder to process
  - Undecidable
  - Spurious ambiguity

Generalized Re-write Rules

- [T NT]* → [T NT]*
- Any number of symbols on either side
- Equivalent to Turing Machines
- Can be intractable
- Can be used to implement a new Android twitter client
    - (Though inefficiently)

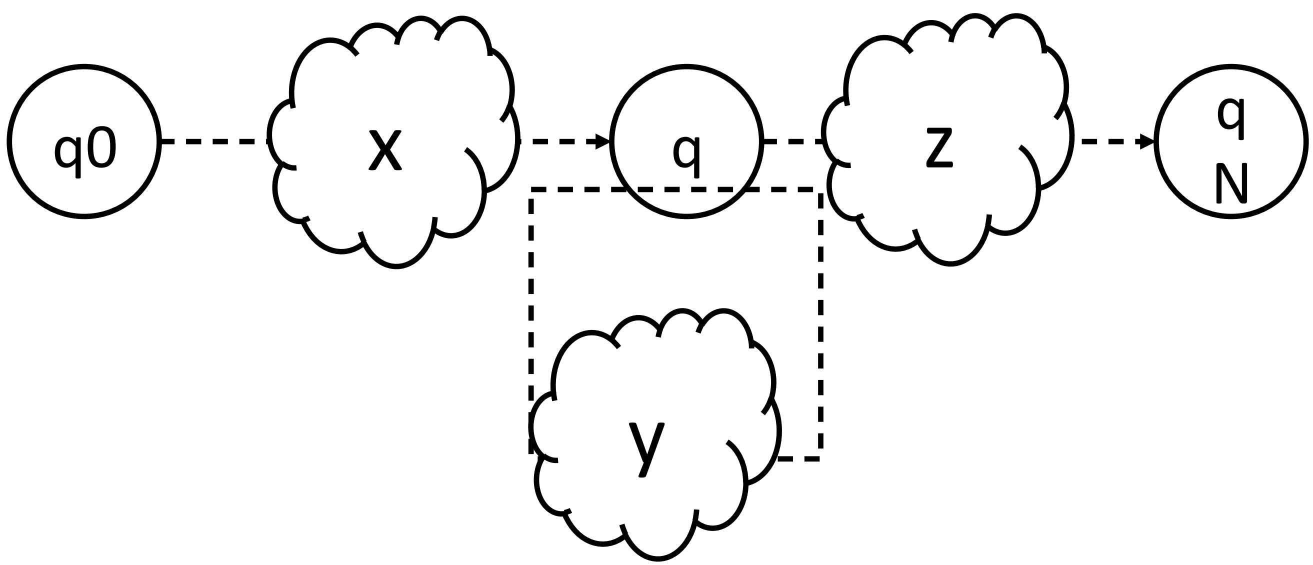
Chomsky Hierarchy



	language class	automaton
0	recursively enumerable	Turing machine
1	context-sensitive	linear bounded automaton
*	mildly context-sensitive	thread automaton
2	context-free	pushdown automaton
3	regular	finite-state automaton

Pumping Lemma

If L is an infinite regular language,
then there are strings x, y, and z
such that $y \neq \varepsilon$ and $xy^nz \in L$, for all $n \geq 0$.



Is English Regular?

L1 =
(the cat|dog|mouse|...)* (chased|bit|ate|...)* likes tuna fish

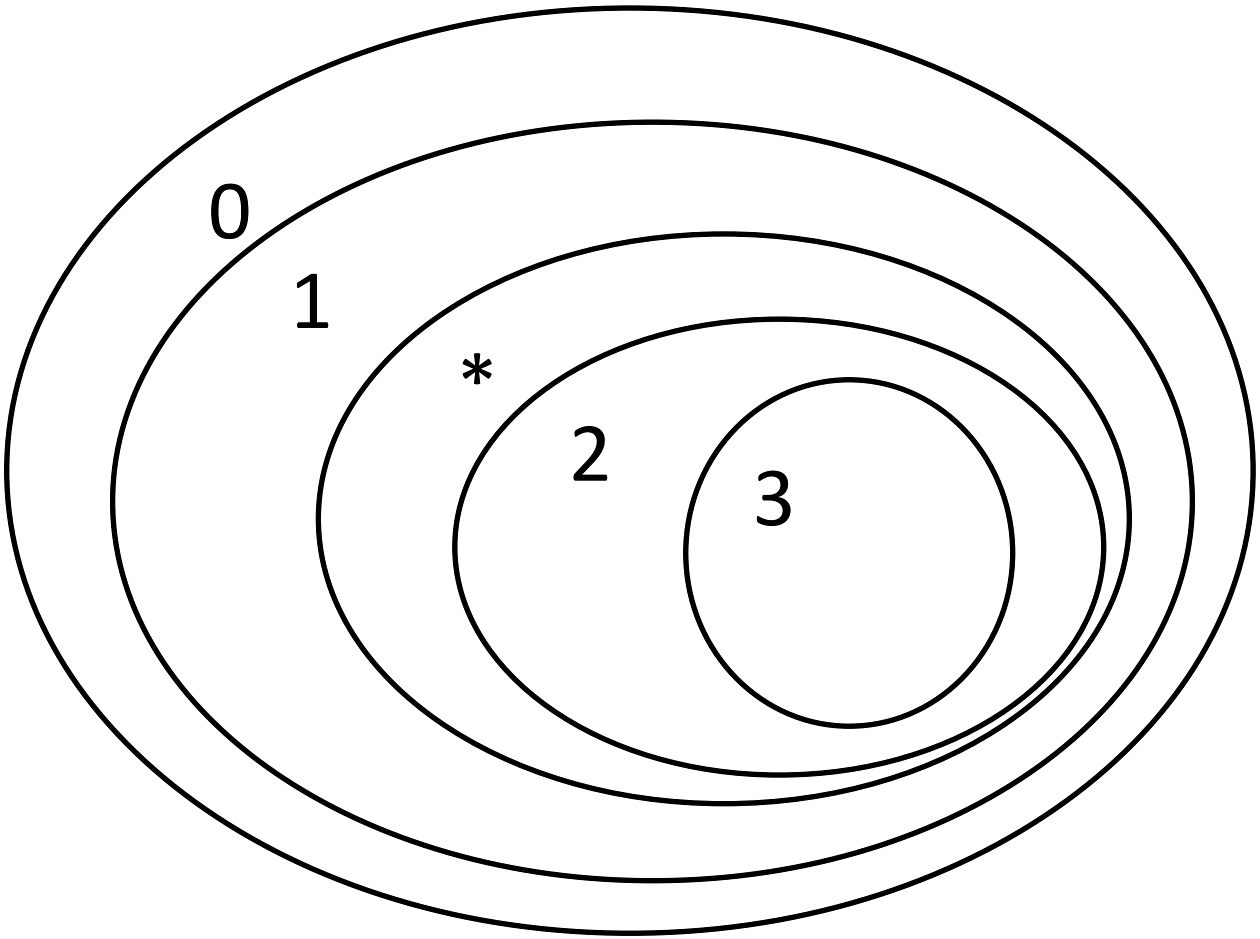
L2 = English

L1 ∩ L2 =
(the cat|dog|mouse|...)n (chased|bit|ate|...)n-1 likes tuna fish

Examples

- The cat likes tuna fish
- The cat the dog chased likes tuna fish
- The cat the dog the mouse scared chased likes tuna fish
- The cat the dog the mouse the elephant squashed scared
- chased likes tuna fish
- The cat the dog the mouse the elephant the

Chomsky Hierarchy



	language class	**automaton**
0	recursively enumerable	Turing machine
1	context-sensitive	linear bounded automaton
*	mildly context-sensitive	thread automaton
2	context-free	pushdown automaton
3	regular	finite-state automaton

Swiss German

dative-Np accusative-Nq dative-taking-Vp accusative-taking-Vq

- Jan säit das mer em Hans es huus hälfed aastriiche]
- Jan says that we Hans the house helped paint
- "Jan says that we helped Hans paint the house"

- Jan säit das mer d'chind em Hans es huus haend wele laa hälfe aastriiche
- Jan says that we the children Hans the house have wanted to let help paint
- "Jan says that we have wanted to let the children help Hans paint the house"

Is Swiss German Context-Free?

L1 =
Jan säit das mer (d’chind)* (em Hans)* es huus haend wele (laa)* (hälfe)* aastriiche

L2 = Swiss German

L1 ∩ L2 =
Jan säit das mer (d’chind)n (em Hans)m es huus haend wele (laa)n (hälfe)m aastriiche

Context Sensitive English

“respectively”

Alice, Bob and Carol will have a beer, a wine and a coffee respectively

A B C … a b c ...

Chomsky Hierarchy

- Natural Language is mildly context sensitive
- But CFGs might be enough
- But RG might be enough
  - If you have very big grammars
  - (and don't really care about parsing)