Chomsky Hierarchy

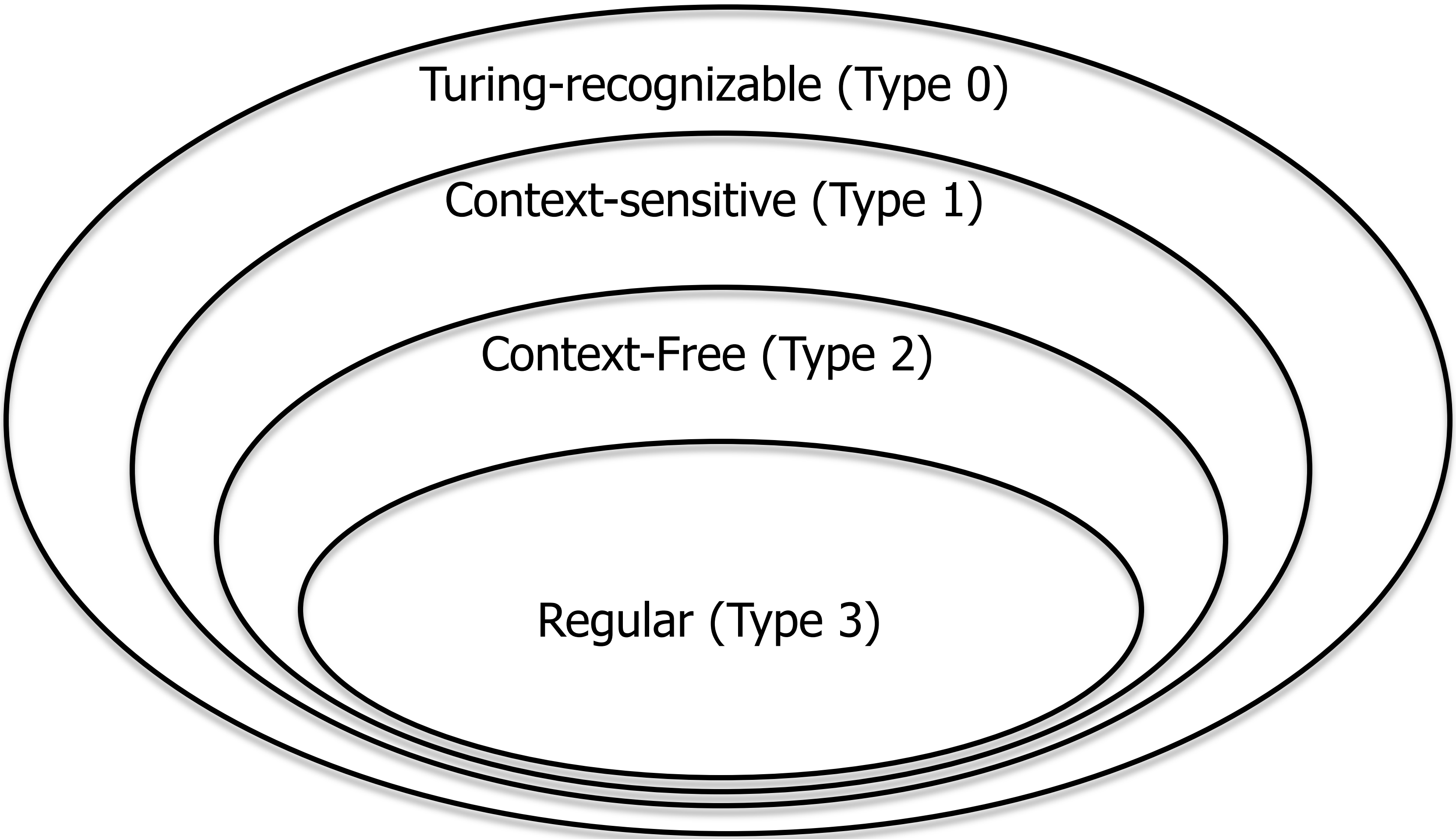
Chomsky Hierarchy

- A containment hierarchy of classes of formal grammars
  - We’ve seen formal grammars used to describe the class of context-free languages
  - It turns out formal grammars can be used to describe other classes of languages we’ve discussed (as well as one we haven’t)

Chomsky Hierarchy

Grammar	Languages	Automaton	Production Rules
Type 0	Turing Recognizable	TM	$\alpha \to \beta$ No restrictions except α contains at least one variable
Type 1	Context-Sensitive	LBA (linear bounded automaton)	$\alpha A \beta \to \alpha \gamma \beta$ α, γ, β all strings, γ must be non-empty, A is a variable
Type 2	Context-Free	PDA	$A \to \gamma$ γ is a non-empty string, A is a variable
Type 3	Regular	DFA	$A \to \alpha$, $A \to \alpha B$ α is a terminal, A,B are variables

Chomsky Hierarchy



Chomsky Hierarchy

- Type 3: Regular Languages
  - Production Rules:
    - A→α
    - A→αB
    - α is a terminal
    - A, B are variables
  - E.g. a*bc*
    - S → aS | bT | b | cU
    - T → cT | c | aU | bU
    - U → aU | bU | cU

Starting rule straight to ε is allowed to generate empty string

Draw the DFA and you'll see the 3 states (S,T,U), and all of the transitions correspond to grammar rules

Chomsky Hierarchy

- Type 2: Context-Free Languages
  - We've studied these.
  - They (in particular the subset of deterministic context-free languages) are the theoretical basis for phrase structure of most programming languages
  - Treat as if in normal form
    - (starting variable straight to ε is allowed for generating the empty string)

Chomsky Hierarchy

- Type 1: Context-Sensitive Languages
  - Production rules:
    - $\alpha A\beta \rightarrow \alpha\gamma\beta$
    - A is a variable
    - Everything else is a string made up of variables and terminals
    - γ must be non-empty
      - This forces $|\alpha A\beta| \leq |\alpha\gamma\beta|$
        - The derivation never shrinks in size
        - (starting variable straight to ε is allowed for generating the empty string)

Chomsky Hierarchy

- Type 1: Context-Sensitive Languages
  - Production rules:
    - $\alpha A\beta \rightarrow \alpha\gamma\beta$
    - Ultimately we want to replace $A \rightarrow \gamma$, but we do it in the context of the surrounding symbols α and β. Thus we can have different rules for replacing A depending on the context.
    - These languages are recognized by a ***linear bounded automaton.***
      - A non-deterministic Turing machine whose tape is bounded by a constant factor times the length of the input

Chomsky Hierarchy

- Type 0: Turing-recognizable Languages
  - Production rules:
    - $\alpha \rightarrow \beta$
    - No restrictions except that α contains at least one variable.
    - Other than that, they are just strings of variables and terminals.
    - Thus it's possible for a production rule to cause the overall derivation to shrink in size!

    - Decidable languages are not a specific member of the overall hierarchy. They would be between Type 0 and Type 1.