What is a Regular Language?

why do we need them?

- Remember: the job of a **scanner/lexer** is to identify the “words” in a program
  - Variable names
  - Keywords
  - Operators
- What we need to do is *define* what those words are
- **Regular expressions** give us the tools to define words: what makes for a valid token

regular expressions

- Regular expressions are a *syntactic tool* for defining *regular languages*
- What is a **language**?
  - A set of strings (words)
  - Composed of symbols (from a finite alphabet)
  - Mathematically: $\mathcal{L} \subseteq \Sigma^*$

  - Key: a language can be *infinite*

“regular” language?

- A language is a (possibly infinite) set of strings
- But there are many different *classes* of languages
  - Language defined by how “complex” it is
- Exact definition is beyond the scope of this class, but roughly, the more complex a language is, the harder it is to:
  - *define* it: what are the rules that determine what strings are in the set
  - *recognize* it: how can we tell whether a particular string is in the set
- Interested in more? See “Chomsky hierarchy”

how will we define regular set?

- An empty set is a regular set: $\emptyset$
- A singleton is a regular set: $S = \{a\}$
- A union of two regular sets is a regular set:

  $S_1 = \{a\}$ $S_2 = \{b\}$ $S_3 = S_1 \cup S_2 = \{a, b\}$
- The concatenation of two regular sets is a regular set:

  $S_1 = \{a, b\}$ $S_2 = \{c, d\}$ $S_3 = S_1 \cdot S_2 = \{ac, ad, bc, bd\}$
- The empty string is a regular set (a language with no words): $S = \{\varepsilon\}$

- More generally: any finite set of strings is a regular set
  - Question: can you prove that from the rules above?

How do we get infinite sets?

- One final operator that gives regular sets their power: **Kleene star**
- Concatenating a regular set 0 or more times is a regular set:
- $S = \{a\}$ $S^* = \{\varepsilon, a, aa, aaa, \ldots\}$
- $S = \{a, b\}$ $S^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots\}$

next: from regular sets to regular expressions

Or: Finally! Regexes!