Introduction
Formal Languages
Regular Languages and Finite State Acceptors
Minimum-State Acceptors

Angluin’s Algorithm
Learning Concepts via Queries
Angluin’s Algorithm
A *formal language* is just a set of strings over some alphabet.

Any computational problem can be rephrased as a question about membership of a string in some formal language.

**EXAMPLE:** Real-World Concept: Properly nested braces in a Java program.

**Abstraction:** “Language of Properly Nested Parentheses”

```
"( ( ( ) ( ) ) )" ∈ L

"( ( ( ( ) ) )" ∉ L
```
More Examples

Let $A$ be the alphabet, for example, $A = \{a, b\}$ (other authors use $\Sigma$ to represent the alphabet)

- Language of all strings over $A$: $\{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$
  [NOTE: $\lambda$ = “the empty string” of length 0]
  This is usually denoted by $A^*$ ("A star")

- All strings of length $\leq 2$: $\{\lambda, a, b, aa, ab, ba, bb\}$

- All strings with at least two $b$s:
  $\{bb, abb, bab, bba, bbb, aabb, abab, baab, abba, \ldots\}$

- All strings that are palindromes:
  $\{\lambda, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, baab, bbbb, \ldots\}$
Finite State Acceptors

Accepts all strings:

Accepts all strings of length $\leq 2$:

Accepts all strings with at least two $bs$:
Regular Sets

Any language that can be accepted by a finite state acceptor is called a *regular language* or *regular set*. Not every language is regular:

- “balanced parentheses” is not regular [can’t “remember” an arbitrary number of left parentheses in finitely many states]
- “palindromes” is not regular [can’t “remember” arbitrarily long first-halves of strings in finitely many states]

Both of these can be accepted by a more powerful machine that uses a stack.
For every regular language $L$, there is a unique finite state acceptor for $L$ that has the fewest possible states. Example:

Both machines accept the same language, but states 1 and 1' are equivalent and can be merged together; so can 2 and 2'. Saying that two states are “not equivalent” means we can perform an experiment to determine which of the two states we are in.
“Witnesses” to Nonequivalence

Given two states in a finite-state acceptor, treat each one as if it is the initial state and find a string that is accepted if we start in one of the two states, but rejected if we start in the other.

![Diagram of states and transitions](image)

States $A$ and $A'$ have “$\lambda$” as a witness string. States $B$ and $B'$ have “$a$” as a witness to their nonequivalence. $C$ and $C'$ are not equivalent as shown by witness string “$aa$,” and $D$ and $D'$ are seen to be nonequivalent because of “$baa$.” Set of witnesses is “suffix-closed.”
Abstract model of a teacher/student relationship:
- Student can ask questions; teacher answers “yes” or “no”
- Student can try to make an “educated guess” about the concept being learned; teacher answers either “yes” or else gives a counterexample that shows why student’s guess doesn’t work

Very severe restrictions ("minimally adequate teacher") — what concepts can be learned, if any?
Basic Ideas Behind Angluin’s Algorithm

- Use “membership queries” to build up information about states
- Use “equivalence queries” to propose a description of a regular set in the form of a finite state machine

States are represented by strings — every input string leads to one and only one state.
If $w$ is a string representing a state, and $A = \{a, b\}$, then $wa$ and $wb$ are strings representing the two possible transitions from that state.
Observation Table

- rows = queried strings
- $S = \text{subset of rows representing states. }$ [NOTE: two rows might represent the same state]
- columns = witness strings
The Algorithm

\[ S \leftarrow E \leftarrow \{\lambda\} \]

construct initial table \( \{S, E, T\} \)

repeat
  while \( \{S, E, T\} \) not closed or not consistent do
    add witnesses to \( E \) and use to split states in \( S \)
    find rows different from rows in \( S \) and add them to \( S \)
  end while

Conjecture a machine \( M \)

if teacher returns a counterexample \( c \) for \( M \) then
  add \( c \) and all its prefixes to \( S \) and update table
end if

until teacher says “yes” to conjectured machine