Automata Theory and Formal Grammars: Lecture 4

Minimal Deterministic Automata

Minimal Deterministic Automata

Last Time:

- Regular Expressions and Regular Languages
- Properties of Regular Languages
- Relating NFAs and regular expressions: Kleene's Theorem

Today:

- Decision procedures for FAs
- Distinguishing Strings with respect to a Language
- Minimum-state DFAs for Regular Languages
- Minimizing DFAs using Partition Refinement

Decision Procedures for FAs

Decision Procedures for FAs

A decision procedure is an algorithm for answering a yes/no question.

A number of yes/no questions involving FAs have decision procedures.

- Given FA M and $x \in \Sigma^*$, is $x \in \mathcal{L}(M)$?
- Given FA M, is $\mathcal{L}(M) = \emptyset$?
- Given FAs M_1 and M_2, is $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$?

Answering the first is easy ... but what about the other two?

Deciding Whether $\mathcal{L}(M) = \emptyset$

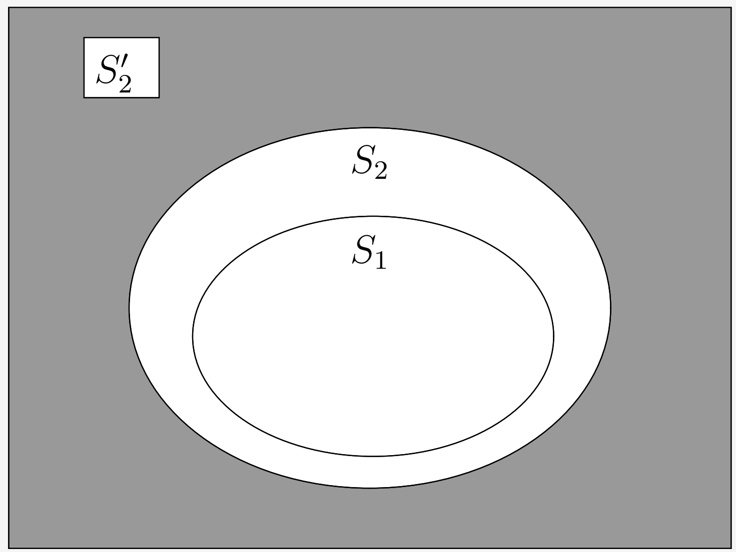
$$\begin{aligned} \mathcal{L}(M) = \emptyset \quad &\iff \quad \forall x \in \Sigma^* .\, x \notin \mathcal{L}(M) \\ &\iff \quad \forall x \in \Sigma^* .\, \delta^*(q_0, x) \notin A \end{aligned}$$

The latter property can be checked using reachability analysis: do all paths from the start state lead to nonaccepting states?

Deciding Whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$

For any sets S_1 and S_2 we can reason as follows.

$$
\begin{aligned}
S_1 \subseteq S_2 \quad &\Longleftrightarrow \quad S_1 - S_2 = \emptyset \\
&\Longleftrightarrow \quad S_1 \cap \overline{S_2} = \emptyset
\end{aligned}
$$



Deciding Whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$ (cont.)

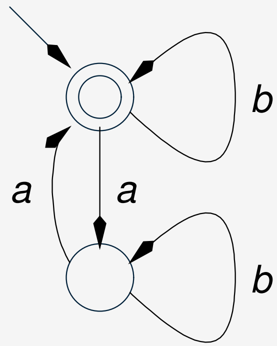
So how can we decide whether or not $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$?

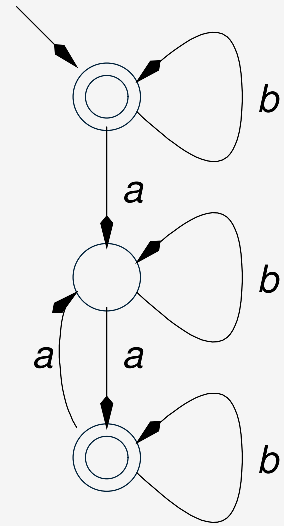
- Build a FA for $\mathcal{L}(M_1) - \mathcal{L}(M_2)$.
  - Complement M_2 to get $\overline{M_2}$.
  - Apply the product construction to get $\Pi(M_1, \overline{M_2})$.
- Check whether or not $\mathcal{L}(\Pi(M_1, \overline{M_2})) = \emptyset$.

Minimizing Automata

How Many States Do You Need in a DFA?

Here are two DFAs recognizing the same language.





The right automaton seems to have a redundant state!

Questions about States in DFAs

- How many states does an DFA need to accept a given language?
- Can a DFA be “minimized” (i.e. can “unnecessary” states be identified and removed)?

We now devote ourselves to answering these questions. All involve a study of the notion of *indistinguishability* of strings.

Indistinguishability

Definition Let $L \subseteq \Sigma^*$ be a langauge. Then the *indistinguishability relation* for L, $\stackrel{L}{\bowtie} \subseteq \Sigma^* \times \Sigma^*$, is defined as follows.

$$x \stackrel{L}{\bowtie} y \text{ iff } \forall z \in \Sigma^*.\, xz \in L \iff yz \in L$$

Intuitively, if $x \stackrel{L}{\bowtie} y$, then any common "extension" to x, y (the "z" in the definition) either makes both xz and yz, or neither, elements of L.

Notes

- $x \stackrel{L}{\bowtie} y$ means x, y are indistinguishable *with respect to language* L. (That is, L must be given in order for $\stackrel{L}{\bowtie}$ to be well-defined.)
- $\stackrel{L}{\bowtie}$ relates arbitrary strings, not just elements in L.
- If $x \in L$ and $x \stackrel{L}{\bowtie} y$ then $y \in L$ also (why?).
- Is it true that $x \in L$ and $y \in L$ imply that $x \stackrel{L}{\bowtie} y$?