NFA → DFA Practice
Analyzing the reduction (cont’d)

- Can reduce any NFA to a DFA using subset alg.
- How many states in the DFA?
  - Each DFA state is a subset of the set of NFA states
  - Given NFA with \( n \) states, DFA may have \( 2^n \) states
    - Since a set with \( n \) items may have \( 2^n \) subsets
  - Corollary
    - Reducing a NFA with \( n \) states may be \( O(2^n) \)
Minimizing DFA: Hopcroft Reduction

- **Intuition**
  - Look to distinguish states from each other
    - End up in different accept / non-accept state with identical input

- **Algorithm**
  - Construct initial partition
    - Accepting & non-accepting states
  - Iteratively refine partitions (until partitions remain fixed)
    - Split a partition if members in partition have transitions to different partitions for same input
      - Two states $x, y$ belong in same partition if and only if for all symbols in $\Sigma$ they transition to the same partition
  - Update transitions & remove dead states

J. Hopcroft, “An $n \log n$ algorithm for minimizing states in a finite automaton,” 1971
Minimizing DFA: Example 1

- DFA

- Initial partitions

- Split partition
Minimizing DFA: Example 1

- **DFA**

- **Initial partitions**
  - Accept \( \{ R \} \) = P1
  - Reject \( \{ S, T \} \) = P2

- **Split partition? \( \rightarrow \) Not required, minimization done**
  - move(S,a) = T \( \in \) P2 \quad \text{← move(S,b) = R \( \in \) P1}
  - move(T,a) = T \( \in \) P2 \quad \text{← move(T,b) = R \( \in \) P1}
Minimizing DFA: Example 3
Minimizing DFA: Example 3

- DFA

- Initial partitions
  - Accept \( \{ R \} = P_1 \)
  - Reject \( \{ S, T \} = P_2 \)

- Split partition? \( \rightarrow \) Yes, different partitions for B
  - \( \text{move}(S,a) = T \in P_2 \)  \(- \text{move}(S,b) = T \in P_2 \)
  - \( \text{move}(T,a) = T \in P_2 \)  \(- \text{move}(T,b) = R \in P_1 \)

DFA already minimal
Minimizing DFA: Example 3
Minimizing DFA: Example 3
Complement of DFA

- Given a DFA accepting language $L$
  - How can we create a DFA accepting its complement?
  - Example DFA
    - $\Sigma = \{a, b\}$

![DFA Diagram]
Reducing DFAs to REs

- General idea
  - Remove states one by one, labeling transitions with regular expressions
  - When two states are left (start and final), the transition label is the regular expression for the DFA
DFA to RE example

Language over $\Sigma = \{0, 1\}$ such that every string is a multiple of 3 in binary
DFA to RE example

Language over $\Sigma = \{0,1\}$ such that every string is a multiple of 3 in binary

\[(0 + 1(0 1^* 0)1)^*\]
Run Time of DFA

- How long for DFA to decide to accept/reject string \( s \)?
  - Assume we can compute \( \delta(q, c) \) in constant time
  - Then time to process \( s \) is \( O(|s|) \)
    - Can’t get much faster!

- Constructing DFA for RE \( A \) may take \( O(2^{|A|}) \) time
  - But usually not the case in practice

- So there’s the initial overhead
  - But then processing strings is fast
Summary of Regular Expression Theory

- Finite automata
  - DFA, NFA

- Equivalence of RE, NFA, DFA
  - RE → NFA
    - Concatenation, union, closure
  - NFA → DFA
    - $\varepsilon$-closure & subset algorithm

- DFA
  - Minimization, complement
  - Implementation