Today's learning goals

- Design NFA recognizing a given language
- Convert an NFA (with or without spontaneous moves) to a DFA recognizing the same language
- Decide whether or not a string is described by a given regular expression
- Design a regular expression to describe a given language
- Convert between regular expressions and automata
Nondeterministic finite automata

- "Guess" some stage of input at which switch modes
- "Guess" one of finite list of criteria to meet
Acceptance in an NFA

An NFA \((Q, \Sigma, \delta, q_0, F)\) accepts a string \(w\) in \(\Sigma^*\) iff we can write \(w = y_1y_2 \cdots y_m\) where each \(y_i \in \Sigma_e\) and there is a sequence of states \(r_0, \ldots, r_m \in Q\) such that:

1. \(r_0 = q_0\)
2. \(r_{i+1} \in \delta(r_i, y_{i+1})\) for each \(i = 0, \ldots, m - 1\)
3. \(r_m \in F\).

\(\delta(r, y)\) is the set of states you can reach from \(r\) on input \(y\).

\(\delta^*(r, w)\) is the set of states you can reach from \(r\) on input \(w\).
NFA
Simulating NFA with DFA

Not quite a closure proof, but …

**Proof:**

**Given** name variables for sets, machines assumed to exist.

**WTS** state goal and outline plan.

**Construction** using objects previously defined + new tools working towards goal. Give formal definition and explain.

**Correctness** prove that construction works.

**Conclusion** recap what you've proved.
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

Proof:

Given a language recognized by NFA \( N = (Q, \Sigma, \delta, q_0, F) \),

WTS there exists a DFA \( M \) with \( L(M) = A \).

Construction

Correctness

Conclusion
Which of the following strings is accepted by this NFA?

A. The empty string
B. 01
C. 111
D. 0
E. More than one of the above.
Subset construction

**Given** A, a language recognized by NFA $N = (Q, \Sigma, \delta, q_0, F)$

**WTS** there exists a DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q'_0, F')$ with

- $Q' = \text{the power set of } Q = \{ X | X \text{ is a subset of } Q \}$
- $q'_0 = \{ \text{states } N \text{ can be in before first input symbol read} \}$
- $F' = \{ \}$
- $\delta' (\text{ }) =$
Which states can this NFA be in before first input symbol is read?

A. q0
B. any state
C. q0, q1
D. q0, q4
E. q0, q1, q4
Subset construction

**Given** A, a language recognized by NFA $N = (Q, \Sigma, \delta, q_0, F)$

**WTS** there exists a DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \} \cup \delta((q_0, \varepsilon)) \cdots \delta^*(q_0, \varepsilon)$
- $F' = \{ \}$
- $\delta' (\text{ }) =$
Subset construction

**Given** A, a language recognized by NFA $N = (Q, \Sigma, \delta, q_0, F)$

**WTS** there exists a DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \} \cup \delta((q_0, \varepsilon))$ …
- $F' = \{ \text{guarantee at least one computation is successful} \}$
- $\delta' ( ) = F' = \{ X \subseteq Q' : X \cap F \text{ not empty} \}$
What does it mean for a set of states X to guarantee at least one computation is successful?

A. X is a subset of F
B. X = F
C. X ∩ F is nonempty
D. X is an element of F
E. None of the above.

From NFA to DFA
Subset construction

**Given** A, a language recognized by NFA $N = (Q, \Sigma, \delta, q_0, F)$  

**WTS** there exists a DFA $M$ with $L(M) = A$  

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with  
- $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$  
- $q_0' = \{ q_0 \} \cup \delta((q_0, \varepsilon))$  
- $F' = \{ X \mid X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$  
- $\delta'( (X, x) ) = \{ \text{union of } \delta(q, x) \text{ for all } q \in X \}$

\[
X = \{ q_1, q_2, q_3 \} \\
\delta(q, x) \cup \delta(q_2, x)
\]
Subset construction

Given $A$, a language recognized by NFA $N = (Q, \Sigma, \delta, q_0, F)$

WTS there exists a DFA $M$ with $L(M) = A$

Construction Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q'$ = the power set of $Q = \{ X \mid X \text{ is a subset of } Q \}$
- $q_0'$ = $\{ q_0 \} \cup \delta((q_0, \varepsilon))$ ...
- $F'$ = $\{ X \mid X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
- $\delta'( (X, x) ) = \{ q \in Q \mid q \text{ is in } \delta(r, x) \text{ for some } r \in X \text{ or accessible via spontaneous moves} \}$
Subset construction examples

DFA
Subset construction examples
Subset construction examples
Recap

Regular sets of strings i.e., those that aren't too complicated are those that are
• the language of some DFA, or equivalently,
• the language of some NFA.

If a set of strings can be expressed as the result of complement, union, intersection, concatenation, Kleene star of regular languages, then it itself is regular.
Regular expressions

- Can all regular languages be expressed as those that are built up from very simple languages using the regular operations?
DFAs vs Regular Expressions

• DFAs – check if a given word is in the language

• Regular Expressions – generate all works in the language
Inductive application of closure

R is a regular expression over $\Sigma$ if

1. $R = a$, where $a \in \Sigma$ \( L(R) = \{a\} \)
2. $R = \varepsilon$ \( L(R) = \{\varepsilon\} \)
3. $R = \emptyset$ \( L(R) = \{\} \)
4. $R = (R_1 \cup R_2)$, where $R_1$, $R_2$ are themselves regular expressions

\[ L(R) = L(R_1) \cup L(R_2) \]
5. $R = (R_1 \circ R_2)$, where $R_1$, $R_2$ are themselves regular expressions

\[ L(R) = L(R_1) \times L(R_2) \]
6. $(R_1^*)$, where $R_1$ is a regular expression

\[ L(R^*) = L(R)^* \]

\( \Sigma \) is shorthand for \((0 \cup 1)\) if $\Sigma = \{0,1\}$, Parentheses may be omitted, $R^+$ means $RR^*$, $R^k$ means $R$ concatenated with itself $k$ times

Watch out for overloaded symbols!
The language described by a regular expression, $L(R)$:

- $L((0 \cup 1) \cup 1) = \{0, 1\}$
- $L((\emptyset \cup \emptyset)) = \{\emptyset\}$
- $L((\Sigma \Sigma \Sigma \Sigma)^*) = \{w \in \Sigma^* \text{ length is a multiple of 4}\}$
- $L(1^* \emptyset 0) = \{\emptyset\}$
- $L(\emptyset ^* \emptyset 0) = \{\emptyset\}$

1. $R = a$, where $a \in \Sigma$
2. $R = \epsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$
5. $R = (R_1 \circ R_2)$
6. $(R_1^*)$
Which of the following strings is not in the language described by

$L = \{013 \mid L_0\{E\} = \{010, E\} ; 5013\}$

$L_0 \{\epsilon\} = \{\}$

$( ( (00)^* (11) ) \cup 01 )^*$

A. 00
B. 01
C. 1101
D. ε
E. I don't know
Let L be the language over \{a,b\} described by the regular expression
\[ ((a \cup \emptyset) \ b^*)^* \]
Which of the following is not true about L?
A. Some strings in L have equal numbers of a's and b's
B. L contains the string aaaaaaa
C. a's never follow b's in any string in L
D. L can also be represented by the regular expression (ab*)^*
E. More than one of the above.
Regular expressions in practice

- **Compilers**: first phase of compiling transforms Strings to Tokens **keywords, operators, identifiers, literals**
  - One regular expression for each token type

- **Other software tools**: grep, Perl, Python, Java, Ruby, …
Theorem: A language is regular if and only if some regular expression describes it.

Lemma 1.55: If a language is described by a regular expression, then it is regular. (can construct DFA/NFA accepting it)

Lemma 1.60: If a language is regular, then it is described by some regular expression