Chap. 1.2 **NonDeterministic Finite Automata** (NFA)

- DFAs: exactly 1 new state for any state & next char
- **NFA**: machine may not work "same" each time
  - More than 1 transition rule for same state & input
    - Any one is valid
    - Choice is made with "crystal ball" – which one will lead to an accepting state if possible
  - Also *ε* (the empty string) is allowed on an edge:
    - State transition can be made without reading any input characters
  - See page 48 Fig. 1.27. two "1s" from q_1 & *ε* on q_2->q_3
    - Accepts all strings from {0,1}* containing 101 or 11
- How does computation proceed? Assume at a step where multiple options are possible – a separate copy of the NFA is started up for each, and run in parallel
  - All with the same starting state and remaining input
  - Each takes a different edge
  - Acceptance if *any* end up in an accepting state
  - See Fig. 1.28 – note a "1" from q1 can go to q2 *or* (because of ε leaving q2) go to q3

- Ways to think of nondeterminism
  - Parallel threads checking different paths
  - Tree of possibilities
  - NFA always "guesses" correctly (crystal ball)
- Examples
  - (p.51) Ex. 1.30 N_2: a "1" in third position *from end*
    - Nondeterminism is knowing when we are 3 symbols from end
  - (p.52) Ex. 1.33 N_3: 0^k, where k is multiple of 2 or 3
    - ε edges lead to two different DFAs
      - One that accepts strings of two 0s
      - One that accepts strings of 3 0s
    - At start, crystal ball "knows" which it is
  - (p.53) Ex. 1.35 N_4: { ε, a, bb, babba, …}

- **(p.53) NFA Formal Definition**: N = (Q, ∑, δ, q0, F)
  - Q, Σ, q0, and F are all as before
  - δ: Q x Σ_ε -> *P*(Q)
    - Σ_ε = Σ U {ε} – epsilon-extended alphabet
    - *P*(Q) = power set of Q – set of all subsets of Q
    - Thus each member of P(Q) is a subset of Q
- N **accepts** w (w a string from Σ*) if
  - w = $y_1 y_2 \ldots y_m$ where y_i ε Σ_ε (i.e. some may be "ε")
  - there exists a sequence of states $r_0, r_1, \ldots r_m$ where
    - r_0 = q0, r_m ε F
    - r_{i+1} ε $\delta(r_i, y_{i+1})$
- p. 54: e.g. N_1 accepts all strings containing 101 or 11
  - Look at transition table – each transition is to a *set* of states
    - Remember ϕ is "empty set"

- (p.55) Theorem Every NFA has an equivalent DFA.
- Proof by construction: given NFA, build matching DFA
- Basic idea: matching DFA has one state for *every possible set of states* that NFA can be in at any time
  - Assume given NFA N = (Q, ∑, δ, q0, F)
  - Build DFA M = (Q', ∑, δ', q0', F')
    - Simple case first, if no ε rules in N
      - Q' = P(Q)
      - q0' = {q0}
      - F' = {R| R in Q', R contains an accept state from F)
      - for each R in Q', and a in Σ:
        - δ'(R, a) = {q|q in Q, for some r in R, δ(r,a)=q}
        - Note: δ'(R, a) can return empty set ϕ
    - If there are ε rules in N: i.e. some δ(q, ε) -> q'
      - Define for any RεQ', E(R) = {q|q ε Q, q can be reached from some q' in R via 0 or more ε edges}
        - **E(R)** = "**ε reachable states**" from R in 0 or more ε
      - Now δ'(R, a) = {q|q in Q, for some r in R, q in E(δ(r,a))}
      - Also q0' = E({q0})
- If NFA has |Q| states, DFA has up to $2^{|Q|}$ states
- **KEY RESULT: NFAs are no more powerful than DFAs!**
  - Just easier to design

- Example 1.41: p. 56 convert NFA N_4 to DFA D
  - Q = {1,2,3} – states of N_4
  - P(Q) = {{},{1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}
    - Each represents a possible state of D
  - Compute E - states reachable by ε - of each state of Q'
    - E({1}) = {1,3} – 3 because of ε from 1 to 3
    - E({2}) = {2} – no ε from 2
    - E({3}) = {3}
    - E({1,2}) = {1,2,3}
    - E{{1,3}) = {1,3}
    - E({2,3}) = {2,3}
    - E({1,2,3}) = {1,2,3}
- Start state is E of N_4's start state 1 = E({1}) = {1,3}
- Accept states are those containing any of N_4's F states ({1})
  - {{1}, {1,2}, {1,3}, {1,2,3}}
- See Fig. 1.43 p. 58
  - Note no edges into {1} or {1,2} so could eliminate
  - See Fig. 1.44 for reduced graph (no way to get to {1} or {1,2}

- Details of Transitions in Fig. 1.43
  - {2} in D
    - input a: {2,3} because N has a edge from 2 to 2 & 3
    - input b: {3}
  - {1} in D
    - input a: ϕ because no a's leave 1 in N
    - input b: {2} because b edge from 1 to 2 in N
  - {3} in D
    - input a: {1,3} because in N a edge from 3 to 1
      - but also from 1 there's an ε edge back to 3
    - input b: ϕ because no a's leave 3 in N
  - {1,2} in D
    - input a: {2,3} while 1 has no a edges, 2 does to {2,3}
    - input b: {2,3} N has a b edge from 1 to 2
      - and a b edge from 2 to 3
  - {1,3} in D
    - input a: {1, 3} while 1 has no a edges,
      - from 3 there is a edge to 1, with an ε back to 3
    - input b: {2} N has a b edge from 1 to 2
      - but no b edges from 3
  - {2,3} in D
    - input a: {1, 2, 3} a edge from 2 to 2,
      - from 3 there is a edge to 1, with an ε back to 3
    - input b: {3} N has a b edge from 2 to 3
      - but no b edges from 3
  - {1,2,3} in D
    - input a: {1, 2, 3} no a edges from 1
      - but a edge from 2 to 2 and 3
      - from 3 there is a edge to 1, with an ε back to 3
    - input b: {2,3} N has a b edge from 1 to 2
      - and b edge from 2 to 3

- Alternative from transition table
- N's original transition table:

State	a	b	ε	E(state)
1	{}	{2}	{3}	{1,3}
2	{2,3}	{3}	{}	{2}
3	{1}	{}	{}	{3}

- D's Transition Table

State	a	b
{1}	E({}) = {}	E(2) = {2}
{2}	E(2)UE(3) = {2}U{3}={2,3}	E(3) = {3}
{3}	E(1) = {1,3}	E({}) = {}
{1,2}	E({}) U E(2) U E(3) = {2,3}	E(2) U E(3) = {2,3}
->{1,3}	E(1) = {1,3}	E(2) U E({}) = {2}
{2,3}	E(1) U E(2) U E(3) = {1,2,3}	E(3) = {3}
{1,2,3}	E(1) U E(2) U E(3) = {1,2,3}	E(2) U E(3) = {2,3}
{}	E({}) = {}	E({}) = {}

- To E's that contain 1 in state, add 3 because of ε 1->3
- Each cell δ'(q',x) is Union of E(δ(q,x)) where q is in set q'
- Red states are in D's final set
- {1,3} is D's start state because its E(1) where 1 is N's state