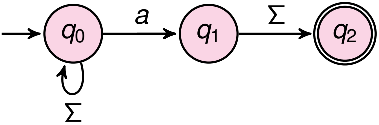
CS 228 : Logic in Computer Science

Krishna. S

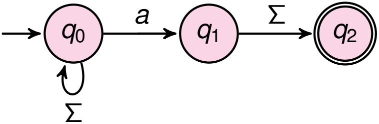
Recap

- FOL over words : Satisfiability
- Translation from formulae φ to equivalent DFA A, $L(\varphi) = L(A)$
- Proof by structural induction, with $\neg, \wedge, \vee$ mapping to complementation, intersection and union
  - Union in DFA–> disjunction in logic
  - Intersection in DFA–> conjunction in logic
  - Complementation in DFA –> Negation in logic
- How to handle quantifiers?

Non-determinism



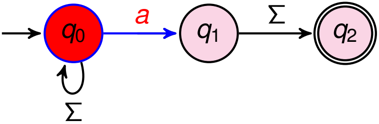
Non-determinism



- Assume we relax the condition on transitions, and allow
  - $\delta : Q \times \Sigma \to 2^Q$
  - $\delta(q_0, a) = \{q_0, q_1\}, \delta(q_2, a) = \delta(q_2, b) = \emptyset$
  - Is *aabb* accepted?

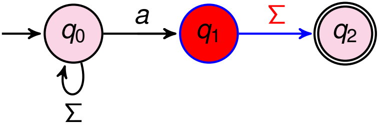
One run of *aabb*

Is *aabb* accepted?



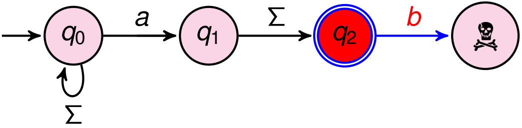
One run of *aabb*

Is *aabb* accepted?



One run of *aabb*

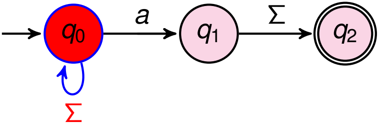
Is *aabb* accepted?



- A non-accepting run for *aabb*

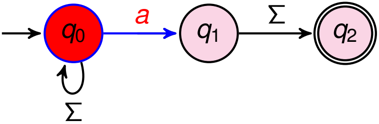
Another run of *aabb*

Is *aabb* accepted?



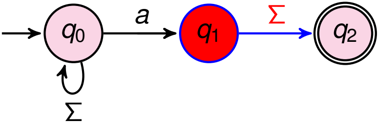
Another run of *aabb*

Is *aabb* accepted?



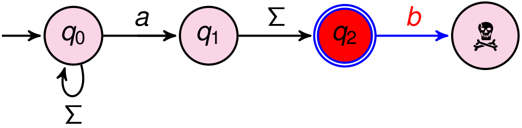
Another run of *aabb*

Is *aabb* accepted?



Another run of *aabb*

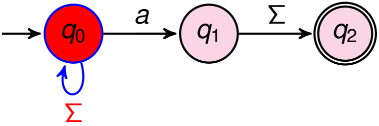
Is *aabb* accepted?



- A non-accepting run for *aabb*

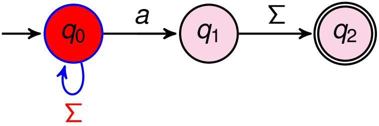
Another run of *aaab*

Is *aaab* accepted?



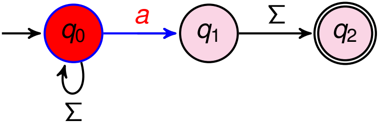
Another run of *aaab*

Is *aaab* accepted?



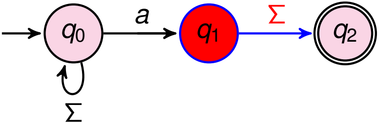
Another run of *aaab*

Is *aaab* accepted?



Another run of *aaab*

Is *aaab* accepted?

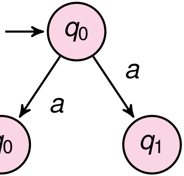


- An accepting run for *aaab*

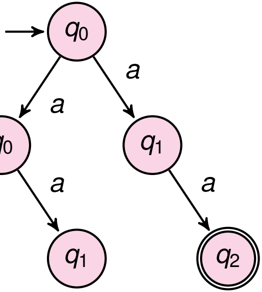
Nondeterministic Finite Automata(NFA)

- $N = (Q, \Sigma, \delta, Q_0, F)$
  - Q is a finite set of states
  - $Q_0 \subseteq Q$ is the set of initial states
  - $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function
  - $F \subseteq Q$ is the set of final states
- Acceptance condition : A word w is accepted iff it has atleast one accepting path

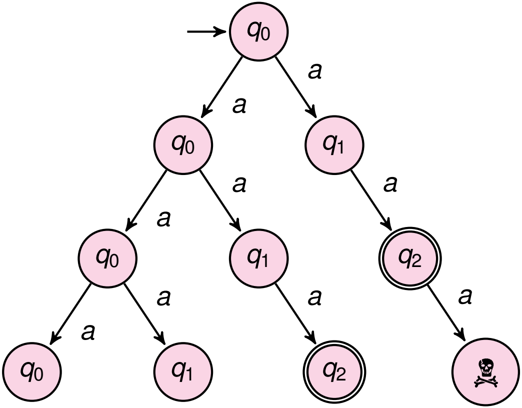
Run Tree of *aaab*



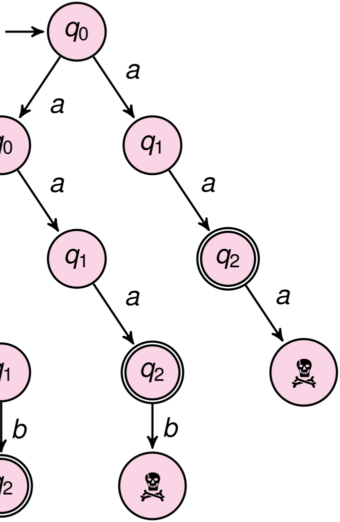
Run Tree of *aaab*



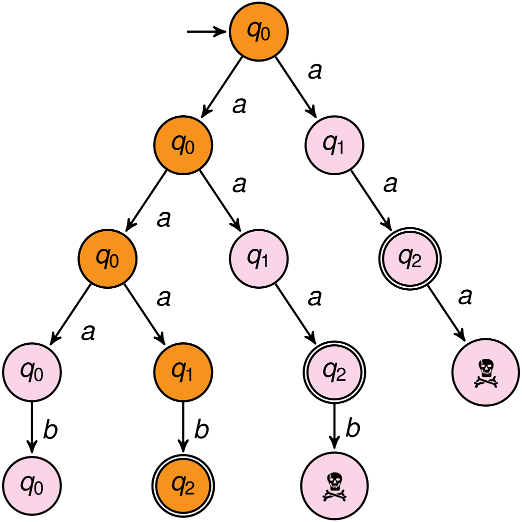
Run Tree of *aaab*



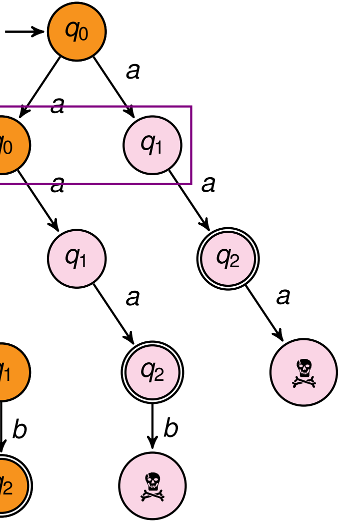
Run Tree of *aaab*



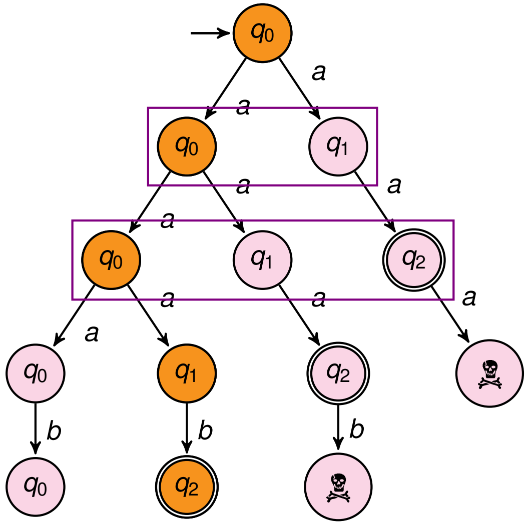
Run Tree of *aaab*



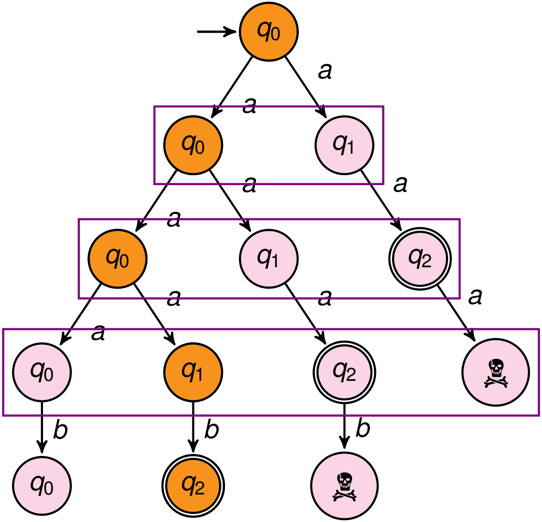
Run Tree of *aaab*



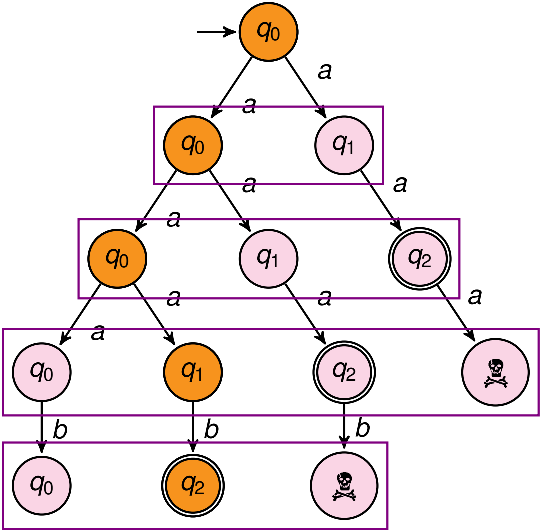
Run Tree of *aaab*



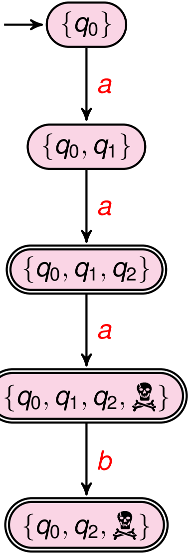
Run Tree of *aaab*



Run Tree of *aaab*

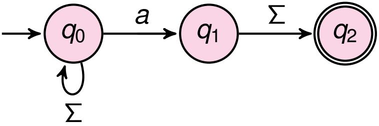


The Single Run



As soon as we get the path which is accepted we are done.

NFA to DFA : On the board



NFA and DFA

- Any DFA is also an NFA

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA
  - Combine states occurring in various runs to obtain a set of states

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA
  - Combine states occurring in various runs to obtain a set of states
  - A set of states evolves into another set of states

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA
  - Combine states occurring in various runs to obtain a set of states
  - A set of states evolves into another set of states
  - Use $\delta : Q \times \Sigma \to 2^Q$, obtain $\Delta : 2^Q \times \Sigma \to 2^Q$

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA
  - Combine states occurring in various runs to obtain a set of states
  - A set of states evolves into another set of states
  - Use $\delta : Q \times \Sigma \rightarrow 2^Q$, obtain $\Delta : 2^Q \times \Sigma \rightarrow 2^Q$
  - Δ is an extension of δ

NFA and DFA

- Any DFA is also an NFA
- Any NFA can be converted into a language equivalent DFA
  - Combine all the runs of w in the NFA into a single run in the DFA
  - Combine states occurring in various runs to obtain a set of states
  - A set of states evolves into another set of states
  - Use $\delta : Q \times \Sigma \to 2^Q$, obtain $\Delta : 2^Q \times \Sigma \to 2^Q$
  - Δ is an extension of δ
  - Accept if the obtained set of states contains a final state

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$

  Union of all the new states after the transition.

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$
Show that

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$ is same as $\hat{\delta} : 2^Q \times \Sigma^* \to 2^Q$ (recall $\delta : Q \times \Sigma \to 2^Q$)

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$
Show that

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$ is same as $\hat{\delta} : 2^Q \times \Sigma^* \to 2^Q$ (recall $\delta : Q \times \Sigma \to 2^Q$)
- $\hat{\Delta}(A, xa) = \Delta(\hat{\Delta}(A, x), a) = \bigcup_{q \in \hat{\Delta}(A,x)} \delta(q, a)$

NFA and DFA

Given NFA $N = (Q, \Sigma, Q_0, \delta, F)$, obtain the DFA $D = (2^Q, \Sigma, Q_0, \Delta, F')$

So see after the transition on individual states of DFS state-set we take the union of reached set. And accepting state is that which has atleast one accepting state from NFA.

- $\Delta : 2^Q \times \Sigma \to 2^Q$ is defined by $\Delta(A, a) = \bigcup_{q \in A} \delta(q, a)$
- $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$

Note that $\hat{\delta}(A, a) = \bigcup_{q \in A} \delta(q, a) = \Delta(A, a)$

Show that

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$ is same as $\hat{\delta} : 2^Q \times \Sigma^* \to 2^Q$ (recall $\delta : Q \times \Sigma \to 2^Q$)
- $\hat{\Delta}(A, xa) = \Delta(\hat{\Delta}(A, x), a) = \bigcup_{q \in \hat{\Delta}(A,x)} \delta(q, a)$
- $\hat{\delta}(A, xa) = \bigcup_{q \in \hat{\delta}(A,x)} \delta(q, a)$

NFA = DFA

NFA and DFA are equivalent if they accept the same language.

$$x \in L(D) \leftrightarrow \hat{\Delta}(Q_0, x) \in F'$$

$$\leftrightarrow$$

$$\hat{\delta}(Q_0, x) \in F'$$

$$\leftrightarrow$$

DFA reaches an accepting state if and only if the NFA also reaches an accepting state on some computation path.

$$\hat{\delta}(Q_0, x) \cap F \neq \emptyset$$

$$\leftrightarrow$$

$$x \in L(N)$$

Regularity

A language L is regular iff there exists an NFA A such that $L = L(A)$