Minimal NFA Problems are hard

Tao Jiang & B. Ravikumar

N Andres Parra

Quick Outline

- Presentation of the problem
- Introduction
- UFA: Unambiguous Finite Automata
- NP-completeness results

Problem:

$A \rightarrow B$

Given a FA of type A, find a *minimum* equivalent FA of type B

DFA $\rightarrow$ DFA

NFA $\rightarrow$ DFA

NFA $\rightarrow$ NFA

DFA $\rightarrow$ NFA ?

Introduction

- Several decision problems for NFA are computationally hard. emptiness and finiteness are exceptions.
- Very few hardness results involving DFA have been shown

Minimal NFA problem:

- Input: FA M of type-A, and integer *k*
- Is there a *k*-state machine of type-B that accepts L(M)?

DFA → DFA

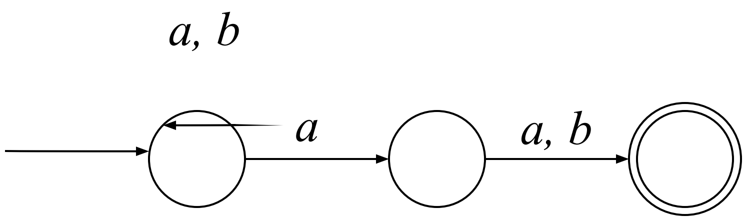
O(nlogn) time -Hopcroft's Algorithm-

NFA → DFA

Exponential lower bound in time and space. PSPACE-hard. NP-hard for unary alphabet

Unambiguous FA (UFA)

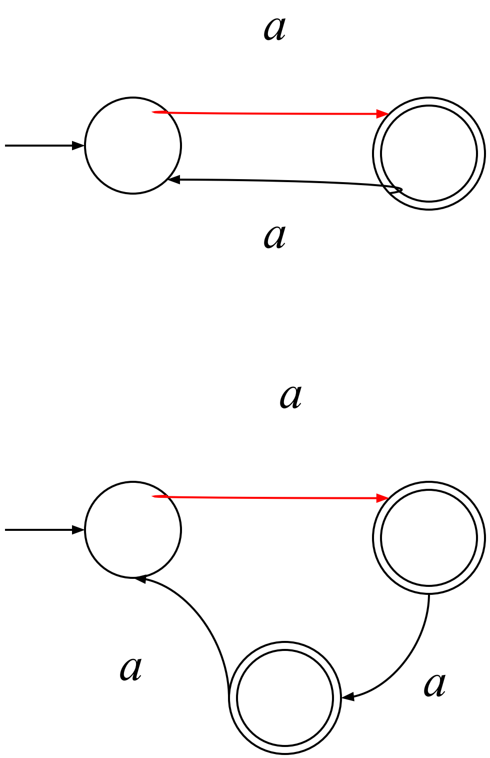
- A UFA is an NFA in which every accepted string has a unique accepting computation



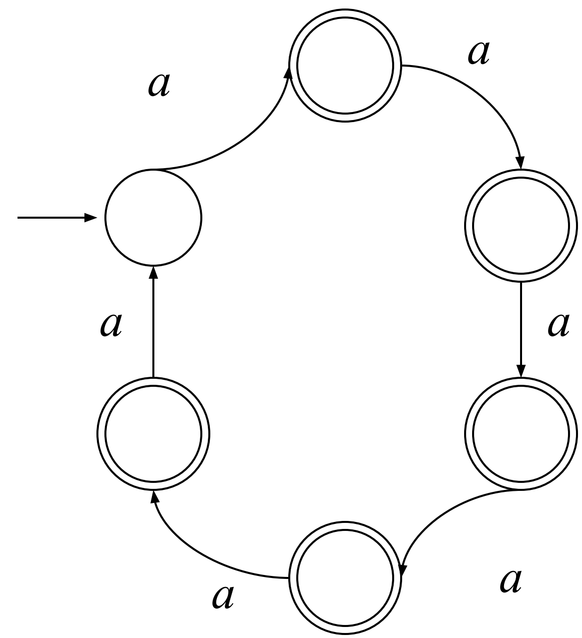
3-state UFA for $\Sigma^* a \Sigma$ with $\Sigma = \{ a, b\}$

- UFA minimization in NP-complete

Example of NFA and UFA for same L



5-state NFA for { a^i | i mod 6 ≠ 0 }



6-state UFA for { a^i | i mod 6 ≠ 0 }

Properties of UFA

- TRAN-SEQ$_M(q,k)$, $q \in$ Q, $k \in$ N:
  Number of state transition sequences of lenght k which take state q into an accepting state.
- ACC-SEQ$_M(k)$, $k \in$ N:
  Number of accepting states transition sequences of length k.
- ACC$_M(k)$, $k \in$ N:
  Number of strings of length k accepted by M

Properties of UFA

- TRAN-SEQ$_M$($q,0$) = 1 if $q \in$ F
  0 otherwise

- TRAN-SEQ$_M$($q,k+1$) =

$$\sum_{t \in Q} d_{q,t} \mathrm{TRAN-SEQ_M}(t,k) \quad d_{q,t} = \left| \left\{ d \in \Sigma \middle| t \in \delta(q,d) \right\} \right|$$

DFA $\rightarrow$ UFA is in NP

- There is a deterministic polynomial-time algorithm for deciding whether the two given UFA, M_1 and M_2 are equivalent
  - Using proper containment based on difference equations of the ACC functions.
- There is a polynomial-time algorithm that, given an NFA M as input, decides whether M is unambiguous
  - Build an NFA M’ that accepts L’, where L’ is the set of strings that can be derived by at least two different accepting paths. Can be done in polynomial time
  - M is unambiguous iff L(M’) is empty.

DFA $\rightarrow$ UFA is NP-complete

- The vertex Cover problem is reduced to the normal set basis problem.
- The Normal set basis problem is reduced to the DFA $\rightarrow$ UFA problem.

- Let C and B be collections of sets. B is said to be a normal basis of C if for each $c \in$ C there is a pairwise disjoint subcollection of B whose union is exactly c.