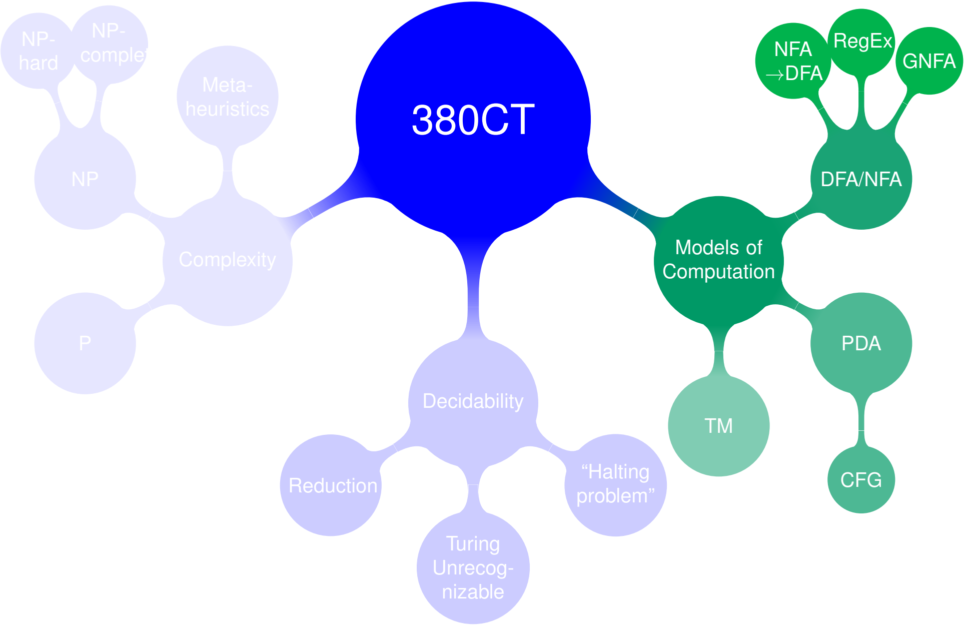
Models of Computation: DFAs & NFAs

Deterministic/Nondeterministic Finite Automata

Dr Kamal Bentahar

School of Engineering, Environment and Computing
Coventry University

10/10/2016


380CT
Models of Computation
DFA/NFA
NFA →DFA
RegEx
GNFA
PDA
CFG
TM
Decidability
Reduction
Turing Unrecog-nizable
“Halting problem”
Complexity
NP
NP-hard
NP-complet
Meta-heuristics
P

A needle in haystack

Given any (finite) haystack, decide whether it contains a needle or not.

- Tedious but doable: simply search every location within the haystack in some order and terminate with **yes** if we find a needle. If we complete the search and no needle was found then terminate with **no**.
- $\rightarrow$ **decision problem**: given data, decide if it has a certain property.
- We may divide all possible instances of the problem into **yes instances** and **no instances** using our process.

Languages

- Simplify the way we describe the problems that machines will solve.
  - Turn *search* problems into *decision* problems
    - Problems with a **yes**/**no** answer.
- **Language recognition**:
  - Think about the English language:
    - Alphabet: $a, b, c, \ldots, x, y, z$ (plus spaces, punctuation, etc.)
    - However, not all strings over this alphabet are members of the language.
      $\rightarrow$ English is a subset of all possible strings over its alphabet.
  - A problem *instance* can be represented as a string of symbols.
  - Instances which yield **yes** are said to *belong* to the corresponding **language** for the problem.
  - Instances which yield **no** (including invalid strings) do not belong to the language.

Language recognition

Decision problems can be encoded as problems of *language recognition*.

Example (Subset sum problem)

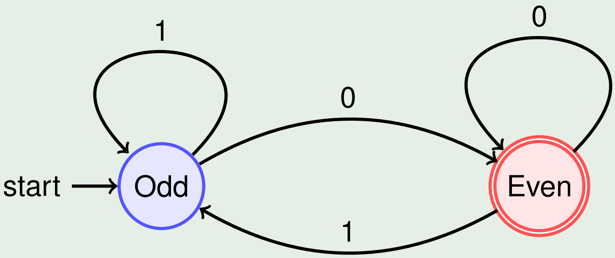
- **Instance:** A set of natural numbers, and a target value.
- **Question:** Is there a subset whose sum equals the target?
- For example, given the set $\{7, 3, 2, 5, 8\}$ and target 10, the answer is **yes** because the subset $\{7, 3\}$ sums to 10.
- This can be represented using the symbols -,{,},0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and comma: $\{7, 3, 2, 5, 8\}10$.
- Given a string using these symbols, we could write a decision procedure to decide if this string belongs to the language of **yes** instances.

Terminology

- Languages are defined over an **alphabet**, denoted by Σ
  Σ is the set of allowable symbols for the language (“*Sigma*”)
- Σ^* denotes the set of all possible strings, over Σ, whose length is finite (“*Sigma star*”)
- A language can be regarded as a subset of Σ^*

The **Deterministic Finite Automaton** (DFA) model

Example (Is a given binary number even?)



The **Deterministic Finite Automaton** (DFA) model

Conceptually, it is a directed, labelled graph which describes how a sequence of symbols from an alphabet will be processed.

- Each node of the graph is called a **state**.
- Each arc of the graph is called a **transition**.
  - The arcs are labelled with symbols from the alphabet.
- We need to designate a *single* **start state**.
- We also need to designate a *set* of **accept states**, which will indicate whether or not the string is accepted after processing.
- Each state must have **exactly one** transition defined for **every** symbol of the alphabet.

Simulation time!

Example

Let us build DFAs over the alphabet $\{a, b\}$ to recognize strings that:

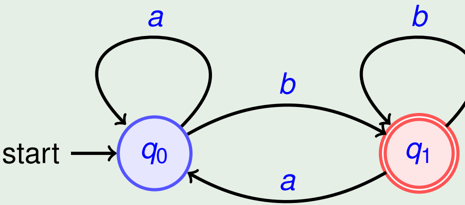
- begin with a
- end with b
- either begin or end with b.
- begin with a and contain at least one b

Definition of a DFA

A *Deterministic Finite Automaton* (DFA) is defined by the 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, F)$ where

- Q is a finite set called the **set of states**
- Σ is a finite set called the **alphabet**
- $\delta \colon Q \times \Sigma \to Q$ is a total function called the **transition function**
- q_{start} is the unique **start state** ($q_{\text{start}} \in Q$)
- F is the **set of accepting states** ($F \subseteq Q$).

Example (DFAs)



This DFA is defined by the 5-tuple $(Q, \Sigma, \delta, q_{start}, F)$ where

- $Q = \{q_0, q_1\}$
- $\delta(state, symbol)$ is given by the table:

	a	b
$\rightarrow q_0$	q_0	q_1
$*q_1$	q_0	q_1

- $q_{start} = q_0$
- $F = \{q_1\}$

Notation

$\delta \colon Q \times \Sigma \to Q$ means that the function *delta* “sends” pairs (q, σ) from $Q \times \Sigma$, made of a *state* q and an *alphabet symbol* σ, to a state in Q.
This is usually given as a table, e.g.

	a	b
$\to q_0$	q_0	q_1
$*q_1$	q_0	q_2
$\vdots$	$\vdots$	$\vdots$

This means that

$$\begin{aligned} \delta(q_0, a) &= q_0 \\ \delta(q_0, b) &= q_1 \\ \vdots &= \vdots \end{aligned}$$

We put “$\to$” next to the start state, and “$*$” next to the accept states.

Notation

2^Q is the "set of subsets of Q", also called the "power set of Q."

Example

$Q = \{A, B, C\}$

Then

$$2^Q = \{ \underbrace{\emptyset}_{\text{Empty set}}, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \underbrace{\{A, B, C\}}_{Q} \}.$$

It has $2^{\text{cardinality of } Q} = 2^{|Q|} = 2^3$ elements.

The **Nondeterministic Finite Automaton (NFA)** model

From the design point of view: NFAs are almost the same as DFAs. Only, instead of every state having **one and only one outward transition** defined **for each symbol**, we may define zero or more transitions for the same symbol.

What if δ is not a total function? $\rightarrow$ NFA

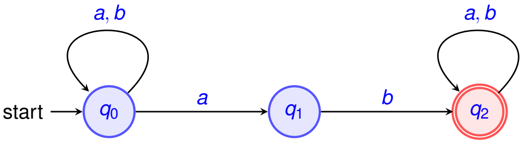
DFA: $\delta \colon Q \times \Sigma \rightarrow Q$ is a **total** function

1. δ is defined for *every* pair (q, σ) from $Q \times \Sigma$
2. δ sends (q, σ) to a **state** from Q (exactly one state, no more, no less)

NFA: $\delta \colon Q \times \Sigma \rightarrow 2^Q$ *may be* a **partial** function

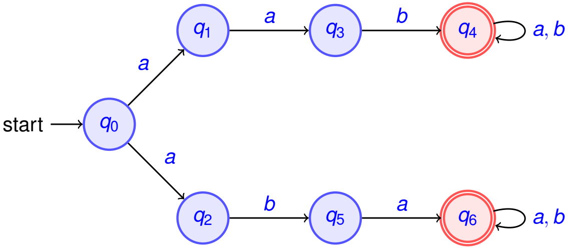
1. δ is *not necessarily* defined for every pair (q, σ) from $Q \times \Sigma$
2. δ sends (q, σ) to a **subset of** Q (many, one, or no states)

NFA example



Q	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\{q_2\}$	$\{q_2\}$

NFA example



Q	a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$
q_1	$\{q_3\}$	$\emptyset$
q_2	$\emptyset$	$\{q_5\}$
q_3	$\emptyset$	$\{q_4\}$
$*q_4$	$\{q_4\}$	$\{q_4\}$
q_5	$\{q_6\}$	$\emptyset$
$*q_6$	$\{q_6\}$	$\{q_6\}$

Simulation time!

Example

Let us build NFAs over the alphabet $\{a, b\}$ to recognize strings that:

- begin with a
- end with b
- either begin or end with b.
- begin with a and contain at least one b

Definition of an NFA

A *Nondeterministic Finite Automaton* (NFA) is defined by the 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called the **set of states**
- Σ is a finite set called the **alphabet**
- $\delta : Q \times \Sigma \to 2^Q$ is a partial function called the **transition function**
- q_0 is the unique **start state** ($q_0 \in Q$)
- F is the **set of accepting states** ($F \subseteq Q$).

Surprise: NFAs recognize exactly the same language as DFAs!