

Models of Computation: DFAs & NFAs

Deterministic/Non-deterministic Finite Automata

Dr Kamal Bentahar

School of Science, Coventry University

Lecture 2

Decision
problems

Languages

Language
recognition
Terminology

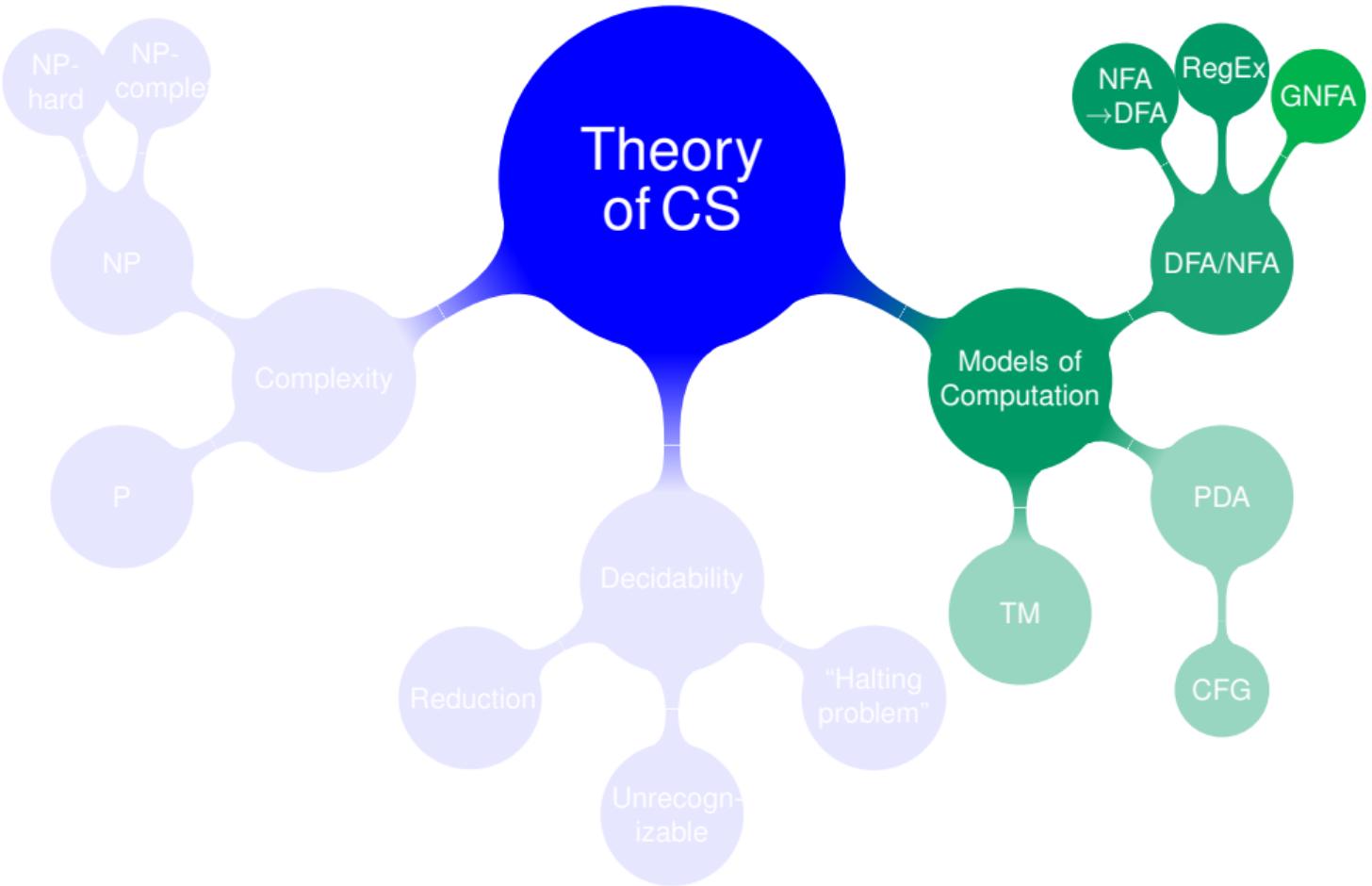
Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP



Last week: We can focus on **decision problems** only.

Decision problems

A yes/no question on a set of inputs.

Given a **search space** and a desired **property**,
decide whether the *search space* contains an item with that *property* or not.

Encoding problems

- The question will be presented as a **string**:

A sequence of symbols from an **alphabet**.

Think about words from the English language:

- Alphabet: $\{a, A, b, B, c, C, \dots, x, X, y, Y, z, Z\}$.
- Example words: Hello, Coventry, and, or, a, ...

- Notation

| Notation | Meaning | Example usage | |
|---------------|---|-----------------------------------|------------------|
| Σ | Alphabet: <u>finite</u> set of symbols . | $\Sigma = \{0, 1\}$ | $\Sigma = \{a\}$ |
| w or s | String made of symbols from Σ | 01100 | aaa |
| $ w $ | Length of the string w | $ 00 = 2$ | $ a = 1$ |
| ε | Empty string – has no symbols. | $ \varepsilon = 0$ | |
| xy | Concatenation of x and y | $x = 0, y = 10 \implies xy = 010$ | |

Concept of “language”

Think about words from the English language again:

- Alphabet: $\{a, A, b, B, c, C, \dots, x, X, y, Y, z, Z\}$.
- However, not all strings over this alphabet are valid words.
In English: *Hello* is valid, but *olleH* is not.
- Divide all possible instances into **yes-instances** and **no-instances**.
- → English is the **set** of “yes-instances over its alphabet.”
- → English is a **subset** of “all possible strings over its alphabet.”

Concept of “language”

In general:

- A (decision) **problem** is a **set of instances** and a required **property**.
- Each problem **instance** is represented by a **string** over an **alphabet Σ** .
- A **yes-instance** satisfies the property required by the problem.
- A **no-instance** does not satisfy the property required by the problem.
- The set of yes-instances defines a **language** associated to the **problem**.

- We say that the yes-instances *belong* to the language.
- No-instances (including invalid strings) *do not belong* to the language.

Language recognition

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

Problem: Is a given number **even**?

Instance: A number n (encoded in binary).

Question: Is n even? (i.e. is it divisible by 2?)

Example

- Given $n = 12_{10} = 1100_2$, the answer is **yes** because $12 = 2 \times 6$.
- Given $n = 13_{10} = 1101_2$, the answer is **no** because $13 = 2 \times 6 + 1$.

Here:

$$\text{Numbers} = \{0, 1, 10, 11, 100, 101, 110, 111, 1000, \dots\}$$

$$\text{Even} = \{0, 10, 100, 110, 1000, \dots\}$$

and

$$\text{Even} \subset \text{Numbers}$$

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Language recognition

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

Problem: Is a given number **even**?

Instance: A number *n* (encoded in binary).

Question: Is *n* even? (i.e. is it divisible by 2?)

- *n* can be represented as a string in binary using only two symbols: 0, 1.
- We can write a **decision procedure** to decide if this string belongs to the language of **yes** instances.

```
1: b  $\leftarrow$  least significant bit of n.  
2: if b = 0 then  
3:   return yes  
4: else  
5:   return no  
6: end if
```

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Terminology

- Languages are defined over an **alphabet Σ** .
- Σ^* : set of all possible strings over Σ , whose **length is finite**.
("Sigma star")

If $\Sigma = \{0, 1\}$ then

$$\Sigma^* = \{ \underbrace{\varepsilon}_{\text{Length 0}}, \underbrace{0, 1}_{\text{Length 1}}, \underbrace{00, 01, 10, 11}_{\text{Length 2}}, \underbrace{000, 001, 010, 011, 100, 101, 110, 111}_{\text{Length 3}}, \dots \}$$

- A language can be regarded as "a subset of Σ^* ".

Example

If $\Sigma = \{0, 1\}$ then the language of even numbers $Even \subset \Sigma^*$ is:

$$Even = \{0, 00, 10, 000, 010, 100, \dots\}$$

Concept of “Model of Computation”

Models of Computation:
DFAs & NFAs

- We want to think more precisely about **problems** and **computation**.
- → categorise them by the **type of computation** which resolves them.
- → idea of **models** of computation:
We introduce simple, theoretical machines and study their limits.
 - Far simpler than Von Neumann Machines, ...
 - ... but some have greater power than Von Neumann machines, ...
 - but cannot be created in reality!
- Our first model is the **Deterministic Finite Automaton** (DFA) model.

Decision problems

Languages

Language recognition
Terminology

Models of Computation

DFAs

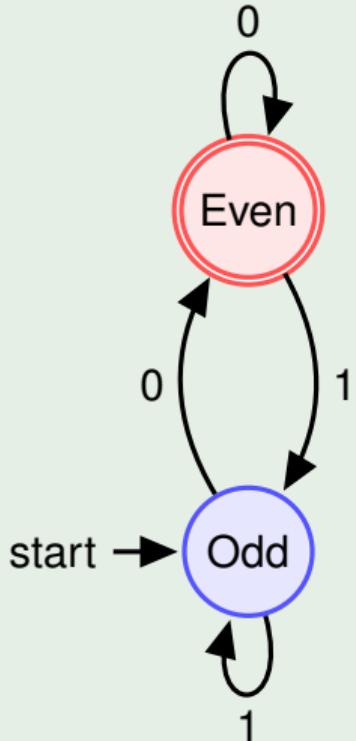
Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

The Deterministic Finite Automaton (DFA) model

Example (Is a given binary number even?)



| Decimal | Binary |
|---------|--------|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

The Deterministic Finite Automaton (DFA) model

A **directed and labelled graph** which describes how a string of symbols from an alphabet will be processed.

- Each vertex is called a **state**.
- Each directed edge is called a **transition**.
 - The edges are labelled with symbols from the alphabet.
- Each state must have **exactly one** transition defined for **every** symbol.
- One state is designated as the **start state**.
- Some states are designated as **accept states**.
- A string is processed symbol by symbol, following the respective transitions:
 - At the end, if we land on an accept state then the string is accepted,
 - otherwise it is rejected.

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Important rules for DFAs

- Each state must have **exactly one transition defined for each symbol**.
- There must be **exactly one start state**.
- There may be **multiple accept states**.
- There may be more than one symbol defined on a single transition.

Example

Let us build DFAs over the alphabet $\{0, 1\}$ to recognize strings that:

- begin with 0;
- end with 1;
- either begin **or** end with 1;
- begin with 1 **and** contain at least one 0.

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP

Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Formal definition of DFAs

Formal 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**.
- $\delta: Q \times \Sigma \rightarrow Q$ 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. $(F \subseteq Q)$

Recall:

- **Total function** means it is defined for “all its inputs.”
- Σ, δ : Sigma, delta. (Greek letters)
- \in, \subseteq : “**element of a set**”, “**subset of a set, or equal**”. (Set notation)

Decision
problems

Languages

Language
recognition
TerminologyModels of
Computation

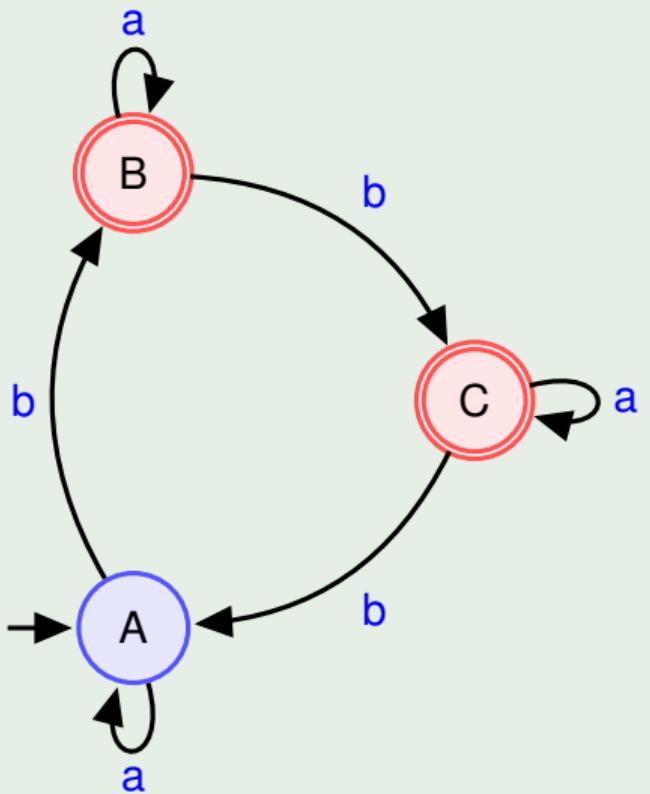
DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Example (Formal specification of a DFA)



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

- $Q = \{A, B, C\}$
- $\Sigma = \{a, b\}$
- δ is given by the table:

| | a | b |
|---------------|---|---|
| \rightarrow | A | A |
| * | B | B |
| * | C | C |

→ indicates the start state
* indicates an accept state.

- $q_{start} = A$
- $F = \{B, C\}$

Notation: Functions/Maps

$\delta: Q \times \Sigma \rightarrow Q$ means that:

- the function δ takes a pair (q, s) as input where:
 - q is a state from Q
 - s is an alphabet symbol from Σ ,
- and returns a state from Q as the result.

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

Table form:

| | a | b |
|-------------------|----------|----------|
| $\rightarrow q_0$ | q_0 | q_1 |
| $*q_1$ | q_0 | q_2 |
| \vdots | \vdots | \vdots |

This means that:

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_0$$

$$\delta(q_1, b) = q_2$$

$$\vdots = \vdots$$

Recall: Power set – set of all subsets

 2^Q is the **set of all subsets of Q** (called: the **power set of Q**)

Example

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

$$2^Q = \left\{ \underbrace{\emptyset}_{\text{Empty set}}, \underbrace{\{A\}, \{B\}, \{C\}}_{\text{One element each}}, \underbrace{\{A, B\}, \{A, C\}, \{B, C\}}_{\text{Two elements each}}, \underbrace{\{A, B, C\}}_Q \right\}.$$

It has 8 elements: $2^{\text{size of } Q} = 2^{\#Q} = 2^3 = 8$.Decision
problems

Languages

Language
recognition
TerminologyModels of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

The Nondeterministic Finite Automaton (NFA) model

From the design point of view: NFAs are almost the same as DFAs.

DFA: every state has one and only one outward transition defined for each symbol.

NFA: every state has zero or more transitions defined for each symbol.

Formally:

DFA: $\delta: Q \times \Sigma \rightarrow Q$ is a **total** function, i.e.

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

NFA: $\delta: Q \times \Sigma \rightarrow 2^Q$ is a **partial** function, i.e.

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

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

Definition of an NFA

A *Nondeterministic Finite Automaton* (NFA) 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**
- $\delta: Q \times \Sigma \rightarrow 2^Q$ is a partial function called the **transition function**
- q_{start} is the unique start state. $(q_0 \in Q)$
- F is the set of accepting states. $(F \subseteq Q)$

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

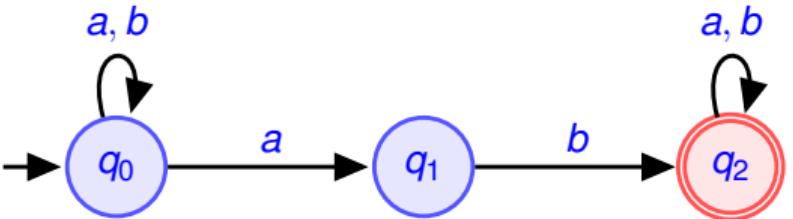
DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples
JFLAP

NFA example



$$Q = \{q_0, q_1, q_2\}$$

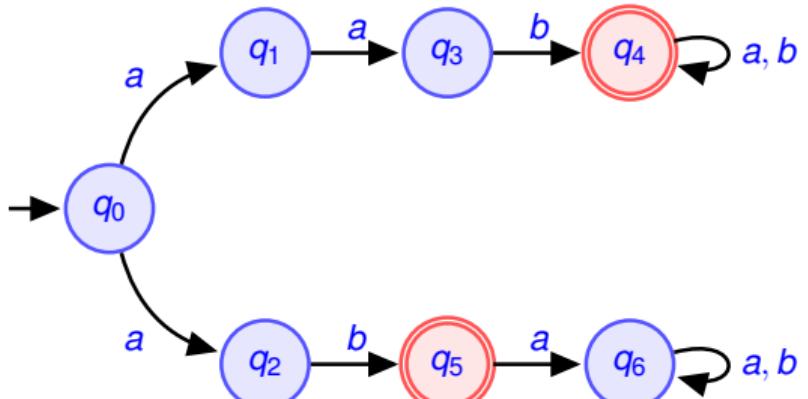
$$\Sigma = \{a, b\}$$

$$q_{\text{start}} = q_0$$

$$F = \{q_2\}$$

| | a | b |
|------------|-------------------|----------------|
| δ : | $\rightarrow q_0$ | $\{q_0, q_1\}$ |
| | q_1 | \emptyset |
| | $*q_2$ | $\{q_2\}$ |

NFA example



$\delta :$

| | 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 | $\{q_6\}$ | $\{q_6\}$ |

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

$$\Sigma = \{a, b\}$$

$$q_{\text{start}} = q_0$$

$$F = \{q_4, q_5\}$$

Example

Let us build DFAs over the alphabet $\{0, 1\}$ to recognize strings that:

- begin with 0;
- end with 1;
- either begin **or** end with 1;
- begin with 1 **and** contain at least one 0.

Decision
problems

Languages

Language
recognition
Terminology

Models of
Computation

DFAs

Example
Informal definition
Important rules
JFLAP
Formal definition
Example
Notation: Functions

NFAs

Power set
Informal description
Formal definition
Examples

JFLAP

Surprise: NFAs recognize exactly the same languages as DFAs!