Classical Information Theory

Violetta Weger

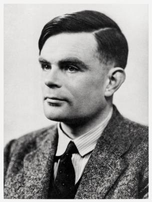
University of Zurich

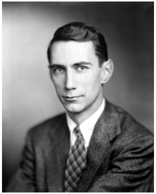
Quantum Information Seminar

13 March 2020

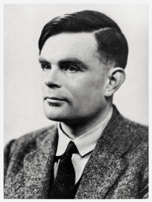
Overview

1. Turing
   1. Automata Theory
   2. Turing Machines
   3. Complexity Classes
2. Shannon
   1. Entropy
   2. Channels





Part 1: Turing



- 1930: Before invention of computers: Turing machines
- Goals:
  - What can Turing machines do and what not → Decidability
  - What can Turing machines do efficiently → Intractability

Ingredients

- Σ the alphabet: finite set of symbols
  $\{a, \ldots, z\}, \{0, 1\}$
- w a word: a finite sequence of symbols in Σ
  $hello, 01101$
- ε the empty word
- $\Sigma^\star$ the Kleene star: set of all possible words with symbols in Σ. More formally:

$$\Sigma^0 = \{\varepsilon\}$$
$$\Sigma^1 = \Sigma$$
$$\Sigma^{i+1} = \{ab \mid a \in \Sigma^i, b \in \Sigma\}$$
$$\Sigma^\star = \cup_{i \geq 0} \Sigma^i$$

- $\mathcal{L} \subseteq \Sigma^\star$ a language: set of words
  $\emptyset, \Sigma^\star$, english

Definition (Deterministic Finite Automaton (DFA))

A deterministic finite automaton A is a tuple $(\Sigma, Q, \delta, q_0, F)$, where

- Σ *is an alphabet*
- Q *is a finite set of states*
- $\delta : Q \times \Sigma \to Q$ *is a transition function*
- $q_0 \in Q$ *is the initial state*
- $F \subseteq Q$ *is the set of final states*

Example

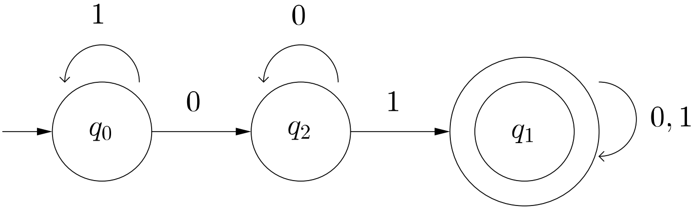
- $\Sigma = \{0, 1\}$
- $Q = \{q_0, q_1, q_2\}$
- transition table

δ	0	1
q_0	q_2	q_0
q_1	q_1	q_1
q_2	q_2	q_1

- $F = \{q_1\}$

Example

transition diagram:



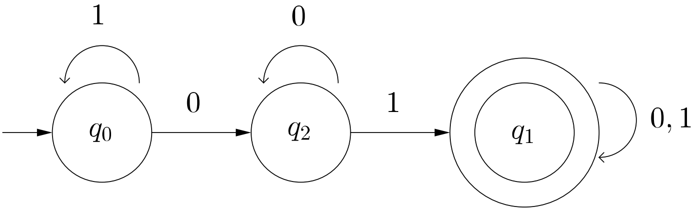
We can define the transition map for words, inductively as follows

$$\begin{aligned}\hat{\delta} : Q \times \Sigma^\star &\to Q \\ (q, wa) &\mapsto \delta(\hat{\delta}(q, w), a).\end{aligned}$$

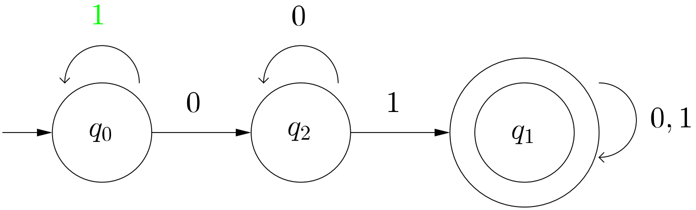
Notions:

- An *execution of a word* $w \in \Sigma^\star$ *by* A is $\hat{\delta}(q_0, w)$.
- A word $w \in \Sigma^\star$ is *accepted by* A, if $\hat{\delta}(q_0, w) \in F$.

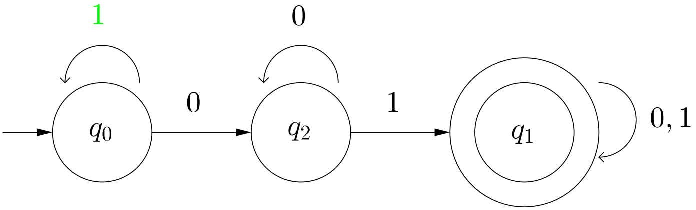
1100 is not accepted by A



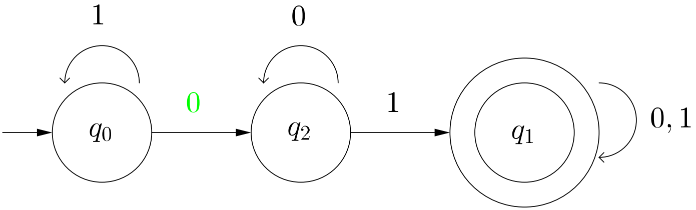
1100 is not accepted by A



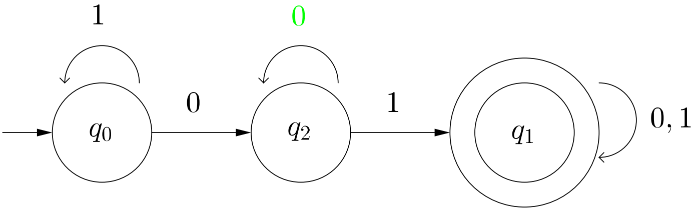
$1{\color{green}1}00$ **is not accepted by** A



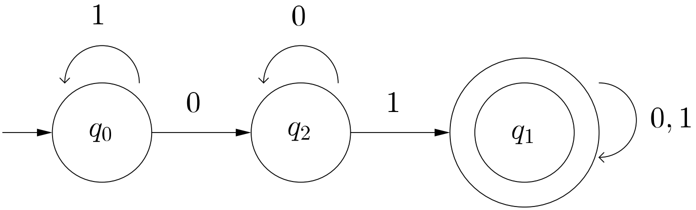
$11\color{green}{0}0$ **is not accepted by** A



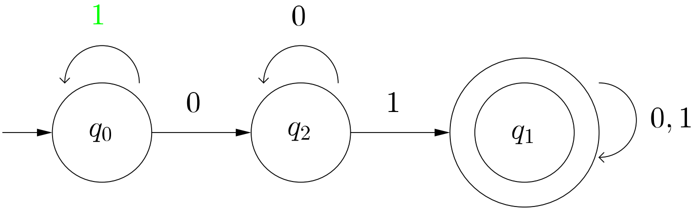
1100 **is not accepted by** A



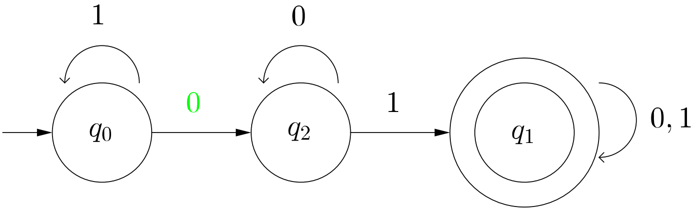
1010 **is accepted by** A



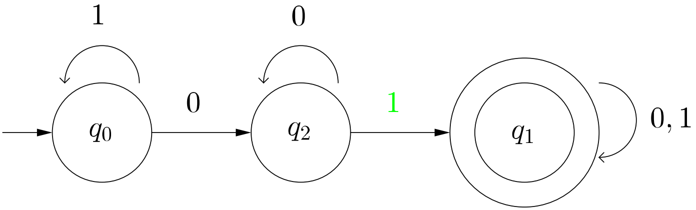
1010 **is accepted by** A



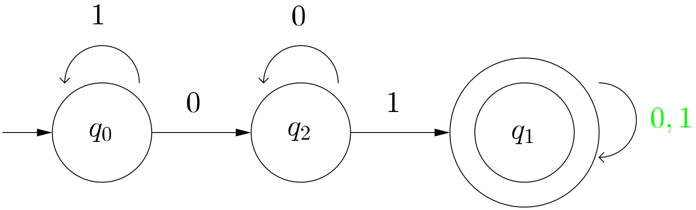
1010 **is accepted by** A



1010 **is accepted by** A



$101\textcolor{green}{0}$ **is accepted by** A



Definition

The language accepted by A is

$$\mathcal{L}(A) = \{w \mid \hat{\delta}(q_0, w) \in F\}.$$

Definition

We call a language $\mathcal{L}$ regular, if there exists a deterministic finite automaton A, such that $\mathcal{L} = \mathcal{L}(A)$.

In our example the language accepted by the automaton is all binary words containing 01.
Notation: $\mathcal{L} = (0+1)^\star 01(0+1)^\star$.
Homework Give a deterministic finite automaton accepting all binary words ending in 00.

Definition (Nondeterministic finite automaton (NFA))

A nondeterministic finite automaton A is a tuple $(\Sigma, Q, \delta, q_0, F)$, *where*

- Σ *is an alphabet*
- Q *is a finite set of states*
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ *is a transition function*
- $q_0 \in Q$ *is the initial state*
- $F \subset Q$ *are the final states*

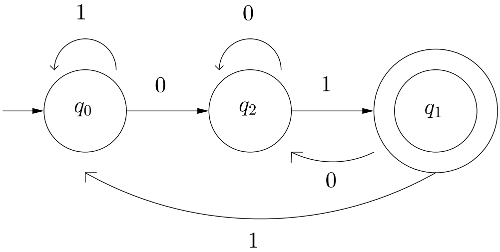
Definition (Nondeterministic finite automaton (NFA))

A nondeterministic finite automaton A is a tuple $(\Sigma, Q, \delta, q_0, F)$*, where*

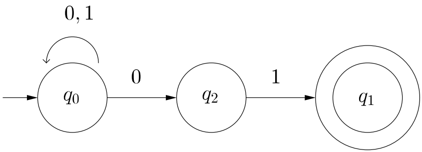
- Σ *is an alphabet*
- Q *is a finite set of states*
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ *is a transition function*
- $q_0 \in Q$ *is the initial state*
- $F \subset Q$ *are the final states*

Automata Theory

DFA accepting words ending in 01



NFA accepting words ending in 01



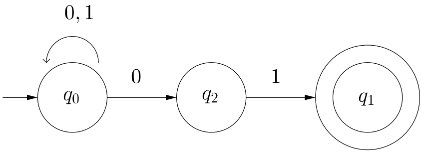
Transition table for the NFA

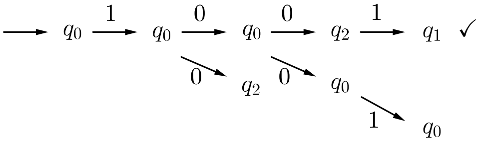
δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\{q_1\}$

and the language accepted by an NFA is

$$\mathcal{L}(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}.$$

Example 1001 is accepted





Theorem (Cool Fact)

If A_N is an NFA, then there exists an A_D a DFA, such that

$$\mathcal{L}(A_N) = \mathcal{L}(A_D).$$

Homework Give a nondeterministic finite automaton accepting all binary words containing 01 or ending in 00.

Theorem (Properties of Regular Languages)

Let $\mathcal{L}, \mathcal{M} \subseteq \Sigma^\star$ be regular languages, then

- $\mathcal{L}^\star$ *is a regular language.*
- $\mathcal{L}\mathcal{M}$ *is a regular language.*
- $\mathcal{L} \cap \mathcal{M}$ *is a regular language.*
- $\mathcal{L} \cup \mathcal{M}$ *is a regular language.*
- $\mathcal{L}^R$ *is a regular language.*
- $\overline{\mathcal{L}}$ *is a regular language.*

Proof of $\overline{\mathcal{L}}$ is a regular language.

$$\overline{\mathcal{L}} = \{w \in \Sigma^\star \mid w \notin L\} = \Sigma^\star \setminus \mathcal{L}.$$

Let $A = (\Sigma, Q, \delta, q_0, F)$ be a DFA accepting $\mathcal{L}$. Define the DFA B to be $(\Sigma, Q, \delta, q_0, Q \setminus F)$. We claim that $\mathcal{L}(B) = \overline{\mathcal{L}}$:

$$w \in \mathcal{L}(B) \Leftrightarrow \hat{\delta}(q_0, w) \in Q \setminus F \Leftrightarrow w \notin \mathcal{L}.$$

$\square$

Homework: Prove that if $\mathcal{L}$ is a regular language, then

$$\mathcal{L}_{pre} = \{w \mid \exists a \in \Sigma \text{ with } wa \in \mathcal{L}\}$$

is a regular language.

Definition (Deterministic Turing Machine (DTM))

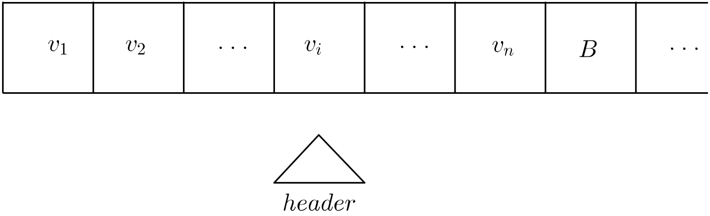
A deterministic Turing machine M is a tuple $(\Sigma, \Gamma, B, Q, q_0, F, \delta)$, where

- Σ *is an alphabet, called input alphabet*
- $\Gamma \supset \Sigma$ *is an alphabet, called tape alphabet*
- $B \in \Gamma \setminus \Sigma$ *is the blank symbol*
- Q *is a finite set of states*
- $q_0 \in Q$ *is the initial state*
- $F \subseteq Q$ *is the set of final states*
- δ *is a partial function*

$$\delta : Q \times \Gamma \to Q \times \{L, R, S\} \times \Gamma$$
$$(q, s) \mapsto (q', D, s')$$

Different notation: $-1 = L$ left, $1 = R$ right, $0 = S$ stay.

Turing Machines



- The tape is bounded on the left.
- The tape is infinite on the right.
- The tape is divided into cells.
- Each cell carries a symbol from Γ.
- The header can read and write.

Definition (Configuration)

A configuration of a DTM $M = (\Sigma, \Gamma, B, Q, q_0, F, \delta)$ *is* (q, i, v), *where*

- $q \in Q$ *is the state in which* M *is in*
- $i \in \mathbb{N}$ *is the cell number to which the header is pointing*
- $v \in \Gamma^\star$ *is the word written on the tape from the first to the last non-blank symbol*

Definition

A configuration $c' = (q', i', v')$ *is derived in one step from* $c = (q, i, v)$ *in the DTM M, if*

- $\delta(q, v_i) = (q', D, a)$,
- $i' = \begin{cases} i+1 & \text{if } D = R \\ i & \text{if } D = S \\ i-1 & \text{if } D = L \end{cases}$,
- $v' = v$, *except that* $v'_i = a$.

Notation: $c \vdash c'$

Definition

A configuration c' is derived from c in the DTM M, if there exists a sequence of configurations $c_1, \ldots, c_k$, such that

$$c \vdash c_1 \vdash \cdots \vdash c_k \vdash c'.$$

Notation: $c \vdash^{\star} c'$.

Notions

- The *inital configuration* of M on the input w is $(q_0, 1, w)$.
- The *execution* of M on the input w is the sequence of configurations $(c_0, \ldots)$, where c_0 is the initial configuration and $c_i \vdash c_{i+1} \forall i$.
- The *final configuration* is a configuration (q, i, v), such that $\delta(q, v_i)$ is not defined.
- The DTM M *stops* on the input w, if the execution of M on the input w reaches a final configuration.
- If the DTM M stops on the input w, then the *computation* $M(w)$ of M on the input w is the word written on the tape, when the final configuration is reached.

- The DTM M *accepts* w, if the execution of M on the input w reaches a final configuration (q, i, v), with $q \in F$.
- The DTM M *rejects* w, if the execution of M on the input w reaches a final configuration (q, i, v), with $q \notin F$.
- The *language accepted by* M is the set of words w, such that M accepts w.
- The *function computed by* M is the partial function, that associated $M(w)$ to w, for all w, such that M stops on w.
- The language $\mathcal{L}$ is *derived* by M, if $\mathcal{L}$ is accepted by M and M always stops.

Example

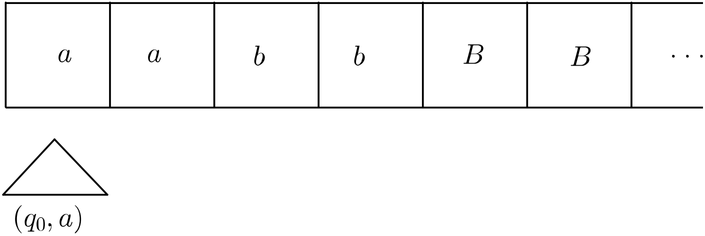
A DTM accepting $\mathcal{L} = \{a^n b^n \mid n \geq 0\}$ is given by

- $\Sigma = \{a, b\}$,
- $\Gamma = \{a, b, D_a, D_b, B\}$,
- $Q = \{q_0, q_{wb}, q_{sa}, q_{fa}, q_e, q_f, q_r\}$,
- $F = \{q_f\}$

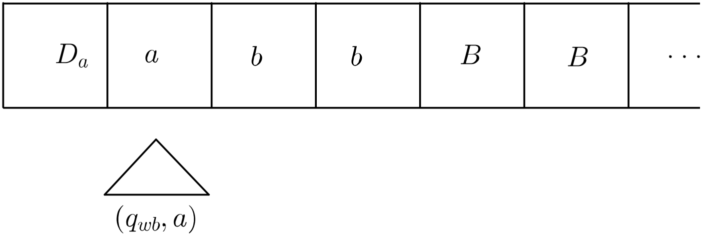
and

δ	a	b	D_a	D_b	B
q_0	(q_{wb}, R, D_a)	(q_r, S, b)			(q_f, S, B)
q_{wb}	(q_{wb}, R, a)	(q_{sa}, L, D_b)		(q_{wb}, R, D_b)	(q_r, S, B)
q_{sa}	(q_{sa}, L, a)		(q_{fa}, R, D_a)	(q_{sa}, L, D_b)	
q_{fa}	(q_{wb}, R, D_a)			(q_e, S, D_b)	
q_e				(q_e, R, D_b)	(q_f, S, B)

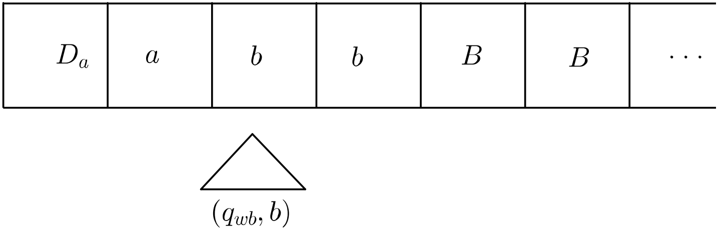
Example $aabb$



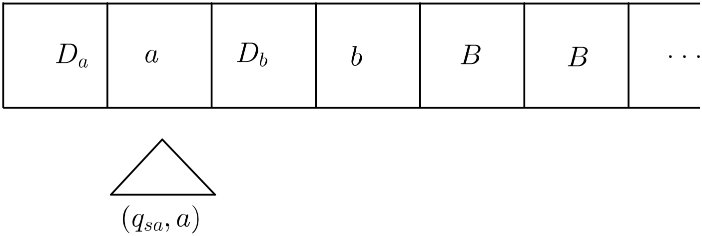
Example $aabb$



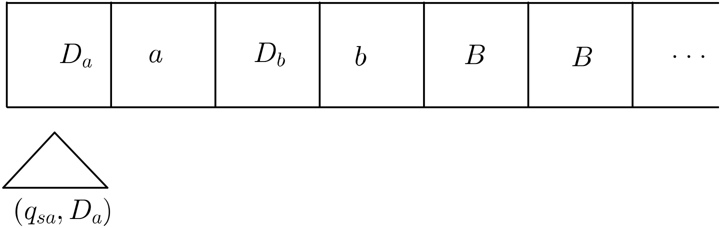
Example $aabb$



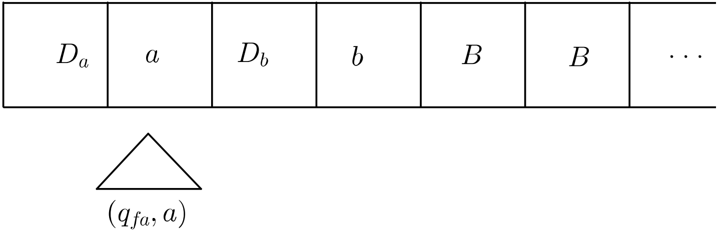
Example $aabb$



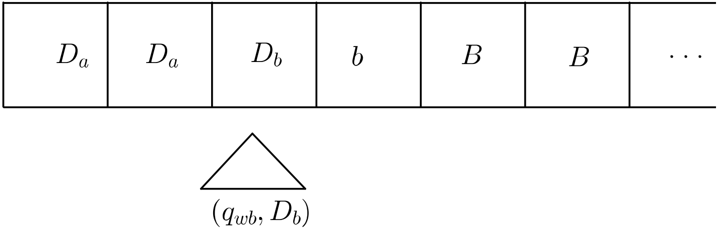
Example $aabb$



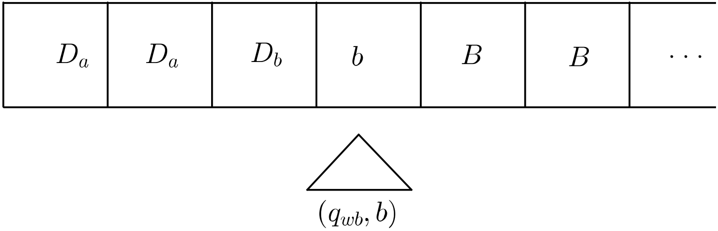
Example $aabb$



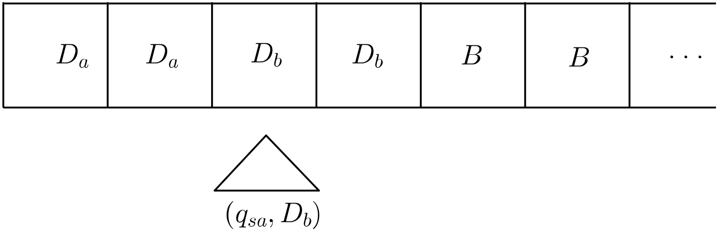
Example $aabb$



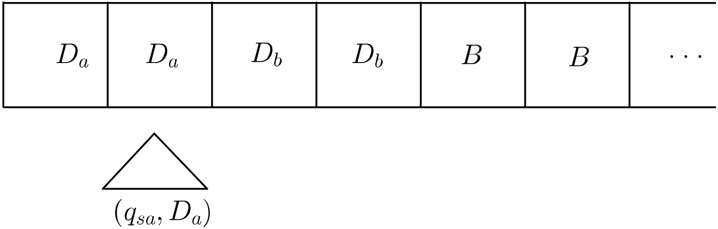
Example $aabb$



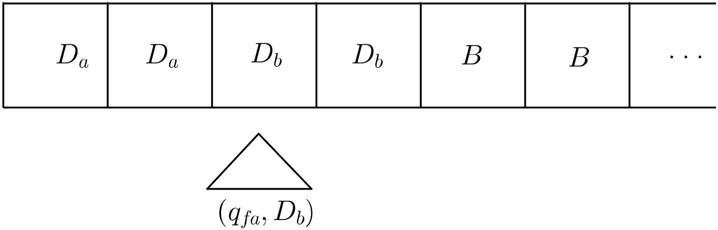
Example $aabb$



Example $aabb$

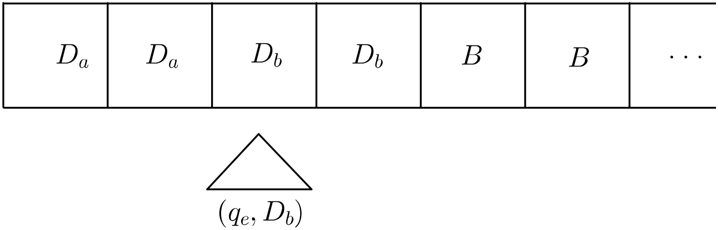


Example $aabb$

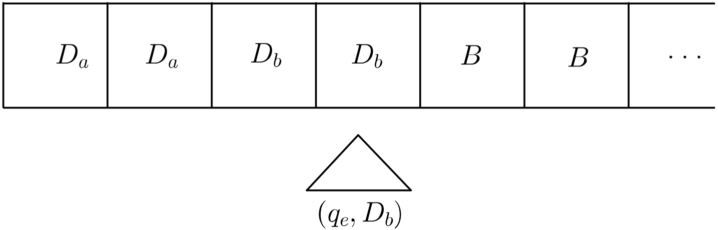


Turing Machines

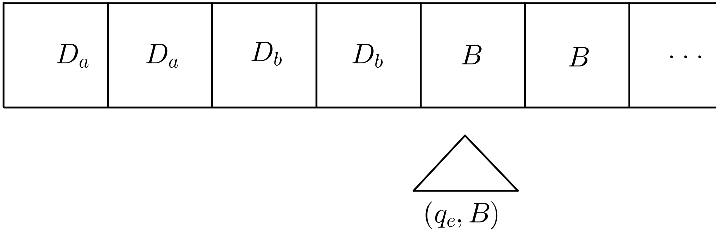
Example $aabb$



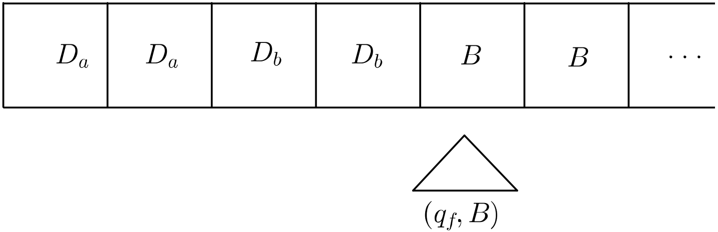
Example $aabb$



Example $aabb$



Example $aabb$



Homework Describe formally a DTM that accepts the binary encodings of even numbers.

Difference to Automaton

- An Automaton is without memory, whereas a TM has a memory in from of the tape.
- The TM can change the word written on the tape.
- An Automaton is a TM, that never changes the direction, nor changes the symbols on the tape.
- TMs accept more languages: recursively enumerable languages

A nondeterministic TM (NTM) is a TM, where the partial function δ has multiple outputs and the TM can choose one.

An NTM M accepts an input w, if there is any sequence of configurations on w that reaches a final configuration.

Theorem

If M_N is an NTM, then there exists M_D a DTM, such that

$$\mathcal{L}(M_n) = \mathcal{L}(M_D).$$

BUT the DTM may take exponentially more time than the NTM.

A nondeterministic TM (NTM) is a TM, where the partial function δ has multiple outputs and the TM can choose one.

An NTM M accepts an input w, if there is any sequence of configurations on w that reaches a final configuration.

Theorem

If M_N is an NTM, then there exists M_D a DTM, such that

$$\mathcal{L}(M_n) = \mathcal{L}(M_D).$$

BUT the DTM may take exponentially more time than the NTM.

A nondeterministic TM (NTM) is a TM, where the partial function δ has multiple outputs and the TM can choose one.

An NTM M accepts an input w, if there is any sequence of configurations on w that reaches a final configuration.

Theorem

If M_N is an NTM, then there exists M_D a DTM, such that

$$\mathcal{L}(M_n) = \mathcal{L}(M_D).$$

BUT the DTM may take exponentially more time than the NTM.

What is the difference between a TM and a classical computer?

- A computer can simulate a TM.
- A TM can simulate a computer (if n is the number of steps of a computer, then the TM needs at most a polnomial in n number of steps)
- They accept the same language

We solved the question of what computers can do. What is it that computers cannot do?

Definition (Decidable)

A language $\mathcal{L}$ is decidable, if there exists a TM M, such that $\mathcal{L} = \mathcal{L}(M)$ and M always stops.

Equivalently, we can ask, are there undecidable languages/problems?

What is the difference between a TM and a classical computer?

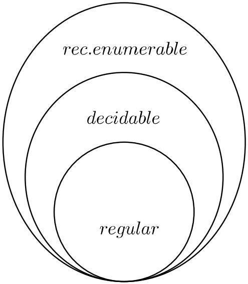
- A computer can simulate a TM.
- A TM can simulate a computer (if n is the number of steps of a computer, then the TM needs at most a polnomial in n number of steps)
- They accept the same language

We solved the question of what computers can do. What is it that computers cannot do?

Definition (Decidable)

A language $\mathcal{L}$ is decidable, if there exists a TM M, such that $\mathcal{L} = \mathcal{L}(M)$ and M always stops.

Equivalently, we can ask, are there undecidable languages/problems?



Examples

- Regular language: Binary words containing 01
- Decidable, but not regular: $\mathcal{L} = \{a^n b^n \mid n \geq 0\}$
- Recursively enumerable but not decidable: the Halting problem: $H(M) = \{w \mid M \text{ halts on input } w\}$, $\mathcal{L} = \{(M, w) \mid w \in H(M)\}$.
- No recursively enumerable $\mathcal{L} = \{M \mid \mathcal{L}(M) = \emptyset\}$.

What can be solved efficiently?

Definition (Running Time)

A TM M is said to have running time/ time complexity $T(n)$, if, whenever M is given an input w of length n, M halts after at most $T(n)$ moves.

Definition (P)

A problem $\mathcal{P}$ is in P, if it can be solved by a DTM in polynomial time.

Examples

- Multiplication: Given $a, b, k \in \mathbb{N}$ encoded in binary, is the kth bit of $a \cdot b$ equal to 1?
- Paths: Given a graph $\mathcal{G}$ and s, t vertices, is there a path from s to t?
- Given $n \in \mathbb{N}$, is n a prime?

Complexity Classes

Definition (NP)

A problem $\mathcal{P}$ is in NP, if it can be solved by a NTM in polynomial time.

or equivalently

Definition

A problem $\mathcal{P}$ is in NP, if a candidate for a solution can be checked by a DTM in polynomial time.

Clearly $P \subseteq NP$ but it remains one of the hardest problems to prove or disprove if $P = NP$.

Examples

- Knapsack: Given $(p_1, \ldots, p_k) \in \mathbb{Z}^k$ and $t \in \mathbb{Z}$, is there a subset $S \subset \{1, \ldots, k\}$, such that $\sum_{i \in S} p_i = t$?
- Clique: Given a graph $\mathcal{G}$ and $k \in \mathbb{N}$, does $\mathcal{G}$ contain a clique of size k, i.e. a set S of k vertices, such that $\forall u, v \in S : (u, v)$ is an edge of $\mathcal{G}$?

Definition (Polynomial Time Reduction)

Given two problems $\mathcal{P}_1$ and $\mathcal{P}_2$, we can reduce $\mathcal{P}_1$ to $\mathcal{P}_2$ in polynomial time, if

- *any instance of $\mathcal{P}_1$ can be transformed in polynomial time to an instance of $\mathcal{P}_2$,*
- *assuming a polynomial time oracle that solves $\mathcal{P}_2$, we get a solution of this instance,*
- *we can transform the solution of $\mathcal{P}_2$ is polynomial time to a solution of $\mathcal{P}_1$.*

$\mathcal{P}_1$ is at least as hard as $\mathcal{P}_2$.

Definition (NP-hard)

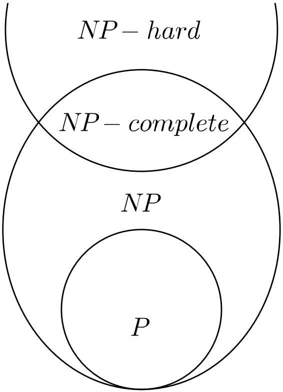
A problem $\mathcal{P}$ is called NP-hard, if any problem in NP can be reduced in polynomial time to $\mathcal{P}$.

Consequences

- Solving an NP-hard problem in polynomial time, means *any* problem in NP can be solved in polynomial time.
- To prove $P = NP$, it is enough to find a polynomial time algorithm for *one* NP-hard problem.
- To prove a new problem is NP-hard, it is enough to find a polynomial time reduction of *one* NP-hard problem to this new problem.

Definition (NP-complete)

A problem $\mathcal{P}$ is called NP-complete, if $\mathcal{P}$ in NP-hard and in NP.



Examples of *NP*-complete problems:

- Knapsack problem
- Clique problem

Examples of problems in *NP*, that are not *NP*-hard:

- Integer factorization: Given $n = p \cdot q \in \mathbb{N}$, where p, q are primes find p and q.
- Discrete logarithm problem: Given $n \in \mathbb{N}$ and $x, y \in \mathbb{Z}/n\mathbb{Z}$, find $k \in \mathbb{N}$, such that $y = x^k \mod n$.

Examples of *NP*-hard problems, that are not in *NP*:

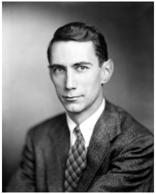
- Halting Problem
- Towers of Hanoi

There are many more complexity classes:
google "Complexity Zoo" to find a list of over 500 classes.

Important Examples

- $PSPACE$: Problems that can be solved by a DTM using polynomial space
- EXP: Problems that can be solved by a DTM in exponential time
- $CO-NP$: the complement of all languages that are in NP.

Part 2: Shannon



- 1948: Father of Information Theory with the article "A mathematical theory of communication"
- Goals:
  - What is "information" $\rightarrow$ Entropy
  - How can we provide information efficiently and reliably? $\rightarrow$ Channels, Codes

Before Shannon in 1928: **Hartley**

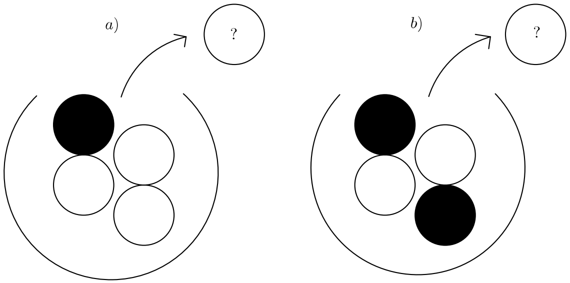
- Information is the value of a random variable
- Also suggested a measure of information

Hartley's measure of the amount of information by observing a discrete random variable X

$$I(X) = \log_b(L),$$

where L is the number of possible values of X.

But there is a problem

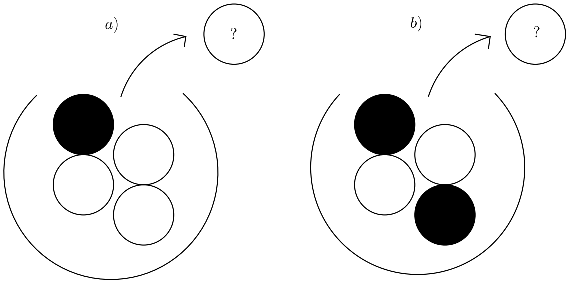


Since $L = 2$, in both examples $I(X) = 1$.

But in a) a white ball is worth less information

Hartley ignores the probabilities of the values

But there is a problem



Since $L = 2$, in both examples $I(X) = 1$.
But in a) a white ball is worth less information
Hartley ignores the probabilities of the values

What should Hartley have done instead?

In a) there is 1 chance out of 4 of choosing a black ball:

$$\log_2\left(\frac{4}{1}\right) = 2$$

and there are 3 chances out of 4 of choosing a white ball:

$$\log_2\left(\frac{4}{3}\right) = 0.415$$

Weight them by their probabilities of occurence:

$$\frac{1}{4}\cdot 2 + \frac{3}{4}\cdot 0.415 = 0.811.$$

Or equivalently

$$-\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right) = 0.811.$$

Entropy

In general, if the ith possible value of X has probability p_i, then the amount of information provided by X is

$$-\sum_{i=1}^{L} p_i \log(p_i).$$

What if $p_i = 0$?

Notation:

- If f is a real valued function, then Supp(f) is the subset of its domain, where f takes non-zero values.
- P_X is the probability distribution for the discrete r.v. X

Definition (Uncertainty/Entropy)

The uncertainty or entropy of a discrete random variable X is

$$H(X) = -\sum_{x \in Supp(P_X)} P_X(x) \log_b(P_X(x)).$$

Entropy

In general, if the ith possible value of X has probability p_i, then the amount of information provided by X is

$$-\sum_{i=1}^{L} p_i \log(p_i).$$

What if $p_i = 0$?

Notation:

- If f is a real valued function, then Supp(f) is the subset of its domain, where f takes non-zero values.
- P_X is the probability distribution for the discrete r.v. X

Definition (Uncertainty/Entropy)

The uncertainty or entropy of a discrete random variable X is

$$H(X) = -\sum_{x \in Supp(P_X)} P_X(x) \log_b(P_X(x)).$$

Remark

$$H(X) = E[-\log(P_X(X))].$$

Also works for discrete random vectors:

Remark

$$H(X, Y) = E[-\log(P_{X,Y}(X, Y))].$$

Example:
X has two possible values x_1 and x_2 with $P_X(x_1) = p$ and $P_X(x_2) = 1 - p$, for some $0 < p < 1$, then the uncertainty of X in bits is the *binary entropy function*

$$H(X) = -p\log_2(p) - (1-p)\log_2(1-p) = h(p).$$

Entropy

Theorem (Information Theory inequality)

For a positive real number r

$$\log(r) \leq (r-1)\log(e).$$

With equality if and only if $r = 1$.

Theorem

If the discrete random variable X *has* L *possible values, then*

$$0 \leq H(X) \leq \log(L),$$

with equality on the left side, if $P_X(x) = 1$ *for some* x*, and equality on the right side, if* $P_X(x) = \frac{1}{L}$ *for all* x*.*

Definition (Conditional Uncertainty)

The conditional uncertainty/entropy of the discrete random variable X given the event $Y = y$ occurs is

$$H(X \mid Y = y) = -\sum_{x \in Supp(P_{X|Y}(.|y))} P_{X|Y} \log(P_{X|Y}(x \mid y)).$$

Remark

$$H(X \mid Y = y) = E[-\log(P_{X|Y}(X \mid Y)) \mid Y = y].$$

Corollary

If the discrete random variable X has L possible values, then

$$0 \leq H(X \mid Y = y) \leq \log(L),$$

with equality on the left side, if $P_{X|Y}(x \mid y) = 1$ for some x, and equality on the right side, if $P_{X|Y}(x \mid y) = \frac{1}{L}$ for all x.

Definition (Conditional Uncertainty)

The conditional uncertainty of the discrete random variable X given the discrete random variable Y is

$$H(X \mid Y) = \sum_{y \in Supp(P_Y)} P_Y(y) H(X \mid Y = y).$$

Remark

$$H(X \mid Y) = E[-\log(P_{X|Y}(X \mid Y))].$$

Corollary

If the discrete random variable X has L possible values then

$$0 \leq H(X \mid Y) \leq \log(L),$$

with equaltiy on the left side, if for all $y \in Supp(P_Y) : P_{X|Y}(x \mid y) = 1$ for some x, i.e. Y essentially determines X, and equality on the right side, if for all $y \in Supp(P_Y) : P_{X|Y}(x \mid y) = \frac{1}{L}$ for all x.

Definition (Information Divergence/ Relative Entropy)

If X and $\tilde{X}$ are discrete random variables with the same set of possible values, then the information divergence between P_X and $P_{\tilde{X}}$ is

$$D(P_X \mid\mid P_{\tilde{X}}) = \sum_{x \in Supp(P_X)} P_X(x) \log\left(\frac{P_X(x)}{P_{\tilde{X}}(x)}\right).$$

Note:

- If there is a $x \in \mathrm{Supp}(P_X)$ but not in $\mathrm{Supp}(P_{\tilde{X}})$, i.e. $P_X(x) \neq 0$ and $P_{\tilde{X}}(x) = 0$, then $D(P_X \mid\mid P_{\tilde{X}}) = \infty$.
- In general: $D(P_X \mid\mid P_{\tilde{X}}) \neq D(P_{\tilde{X}} \mid\mid P_X)$.

Remark

$$D(P_X \,||\, P_{\tilde{X}}) = E\left[\log\left(\frac{P_X(x)}{P_{\tilde{X}}(x)}\right)\right].$$

Theorem (Divergence Inequality)

$$D(P_X \,||\, P_{\tilde{X}}) \geq 0,$$

with equality if and only if $P_X = P_{\tilde{X}}$.

Knowing Y reduces our uncertainty about X

Theorem (2. Entropy Inequality)

For any two discrete random variables X, Y

$$H(X \mid Y) \leq H(X),$$

with equality if and only if X *and* Y *are independent.*

Theorem (The Chain Rule for Uncertainty)

$$H(X_1, \ldots, X_N) = H(X_1) + H(X_2 \mid X_1) + \cdots + H(X_N \mid X_1, \ldots, X_{N-1}).$$

But wait, what is information now?

Shannon: "Information is the difference between uncertainties."
How much information does the random variable Y give about the random variable X?
Shannon: "The amount by which Y reduces the uncertainty about X."

Definition (Mutual Information)

The mutual information between the discrete random variable X and Y is

$$I(X;Y) = H(X) - H(X \mid Y).$$

But wait, what is information now?

Shannon: "Information is the difference between uncertainties."
How much information does the random variable Y give about the random variable X?
Shannon: "The amount by which Y reduces the uncertainty about X."

Definition (Mutual Information)

The mutual information between the discrete random variable X and Y is

$$I(X;Y) = H(X) - H(X \mid Y).$$

But wait, what is information now?

Shannon: "Information is the difference between uncertainties."
How much information does the random variable Y give about the random variable X?
Shannon: "The amount by which Y reduces the uncertainty about X."

Definition (Mutual Information)

The mutual information between the discrete random variable X and Y is

$$I(X;Y) = H(X) - H(X \mid Y).$$

Why mutual?

$$
\begin{aligned}
H(X, Y) &= H(X) + H(Y \mid X) \\
&= H(Y) + H(X \mid Y)
\end{aligned}
$$

Hence

$$H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

That is

$$I(X; Y) = I(Y; X).$$

Definition (Conditional Mutual Information)

The conditional mutual information between the discrete random variable X and Y given the event $Z = z$ occurs is

$$I(X; Y \mid Z = z) = H(X \mid Z = z) - H(X \mid Y, Z = z).$$

Definition (Conditional Mutual Information)

The conditional mutual information between the discrete random variable X and Y given the discrete random variable Z is

$$I(X; Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z).$$

Theorem

For any two discrete random variables X, Y

$$0 \leq I(X;Y) \leq \min\{H(X), H(Y)\},$$

with equailty on the left side, if X and Y are independent, and equality on the right side, if Y essentially determines X or X essentially determines Y.

Channels

Now we have solved the question of what is information.

How can we transmit information efficiently and reliably from its source to the destination?

- The source can choose the signal.
- The channel specifies the conditional probabilites of the signals that can be received.

Channels

Now we have solved the question of what is information.

How can we transmit information efficiently and reliably from its source to the destination?

- The source can choose the signal.
- The channel specifies the conditional probabilites of the signals that can be received.

Channels

We will only consider time-discrete channels, such that the channel input and output can be described as sequences of random variables:

- Input sequence: $X_1, \ldots$
- Output sequence: $Y_1, \ldots$

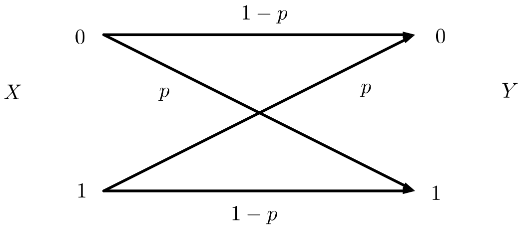
Definition (Discrete Memoryless Channel (DMC))

A discrete memoryless channel (DMC) consists of

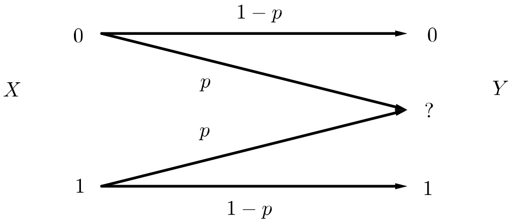
- *A the input alphabet: its symbols represent one of the signals the sender chan choose*
- *B the output alphabet: its symbols represent one of the output signals*
- *$P_{Y|X}(. \mid x)$ the conditional probability distribution over B for all $x \in A$, which governs the channel behaviour, such that*

$$P(y_n \mid x_1, \ldots, x_n, y_1, \ldots, y_{n-1}) = P_{Y|X}(y_n \mid x_n).$$

Example: Binary Symmetric Channel (BSC)



Example: Binary Erasure Channel (BEC)



Channels

Definition (DMC without Feedback)

We call a DMC to be without feedback, if

$$P(x_n \mid x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1}) = P(x_n \mid x_1, \ldots, x_{n-1}),$$

i.e., we are not using the past output digits to choose new inputs.

Theorem

When a DMC is used without feedback, then

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} P_{Y|X}(y_i \mid x_i).$$

Recall: The DMC specifies the conditional probability distribution, but the sender is free to choose the input probability distribution.

Definition (Capacity)

The capacity of a channel is

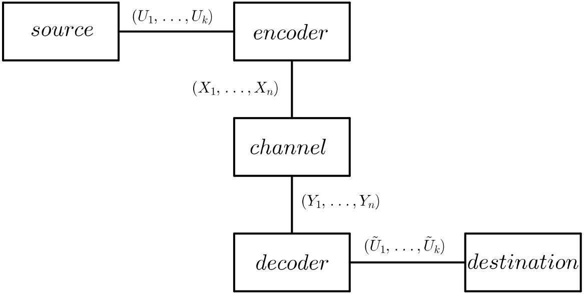
$$C = \max_{P_X} \{ I(X;Y) \}.$$

Example:

- BSC: $C = 1 - h(p)$
- BEC: $C = 1 - p$

How to reliably transmit information through a DMC?

We use k *information bits* to encode a message into n *channel digits*.
This has a rate of $R = \frac{k}{n}$ bits per use.
The channel is noisy, i.e., it enters some errors in what we send:
We encode $U_1, \ldots, U_k$ and send this to a receiver, while the receiver might decode $\tilde{U}_1, \ldots, \tilde{U}_k$.



Definition (Bit Error Probability)

The fraction of the digits that are in error is the bit error probability

$$P_b = \frac{1}{k}\sum_{i=1}^{k} p_{ei},$$

where

$$p_{ei} = P(\tilde{U}_i \neq U_i).$$

Definition (Block Error Probability)

The block error probability

$$P_B = P((\tilde{U}_1, \ldots, \tilde{U}_k) \neq (U_1, \ldots, U_k)).$$

Clearly

$$P_b \leq P_B \leq kP_b.$$

Theorem

If the information bits are sent at rate R via a DMC of capacity $C < R$ without feedback, then the bit error probability at the destination satisfies

$$P_b \geq h^{-1}\left(1 - \frac{C}{R}\right),$$

where h is the binary entropy function, and

$$h^{-1}(x) = \min\{p \mid h(p) = x\}.$$

Thus P_b cannot be very small when $R > C$.

Theorem (Noisy Coding Theorem for DMC)

Consider a transmission of information bits at rate $R = \frac{k}{n}$ *via a DMC of capacity* $C > R$ *without feedback, then given any* $\varepsilon > 0$ *one can always achieve*

$$P_B < \varepsilon$$

by choosing n *large enough.*

This was the bombshell of Shannons 1948 paper:

If $R < C$ *one can get reliability.*

Summary

- What computers can do:
  - Automata theory are memoryless Turing machines, accepting regular languages
  - Turing machines are basically classical computers
- How efficiently they can do it:
  - Complexity classes
- What is information:
  - the difference of uncertainty
- How to transmit information reliably:
  - through channels
  - using coding theory

The End

References

- Part 1: Turing
  - My memory on Mathilde Bouvels lecture "Computability and Complexity Theory"
  - "An Introduction to Automata Theory, Languages and Computation" by John Hopcroft
- Part 2: Shannon
  - Lecture notes on "Applied Digital Information Theory" by James L. Massey