Discrete Mathematics 2

Dedicated to Alan Turing

November 18, 2020

1 Axiomatic Systems and Complexity - Thilo - 6 lectures

1.1 Historical Introduction

- **Fermat's little theorem:** For every integer a, and every prime p, the number $a^p - a$ is an integer multiple of p. E.g. $2^3 - 2 = 6 = 3 \times 2$
- Euclid's postulates (300BC) - geometric axioms. 1. A straight line can be drawn between 2 points; 2. Any line segment can be extended indefinitely into a straight line; 3. A circle can be drawn from a line segment; 4. All right angles are congruent. 5. Parallel postulate
- In the 19th century more "axiomisation" of numbers took place leading to a union of logic and mathematics.
- Set theory invented by Cantor and Dedekind? Russel and Zermelo come along and find the maths to be flawed.
- Russel and Hilbert try to put down more concrete mathematical building blocks in the early 20th century. Hilbert's programme.
- Gödel's incompleteness theory shows that every axiomatic system contains truths that cannot be proven (**1st incompleteness theorem**) and that no axiomatic system can prove its own consistency (**2nd incompleteness theorem**) $--> $ No free lunch - we need assumptions to formulate theorems and Garbage in, Garbage out - we cannot be sure about the consistency of axioms.
- Alan Turing formulated similar statements in relation to computation.**Computability** - some solutions cannot be effectively computed. **Halting Problem** - There cannot be a general algorithm that can determine whether a given program given an input will terminate.
- Von Neumann went on to develope the first modern computer based of Turing's work and invent Game Theory.

1.2 Numbers

1.2.1 Set Theory

- Notation: $\subset$ - PROPER subset, $\subseteq$ - SUBSET, $|$ - such that (instead of :), $\supset$ - PROPER SUPERset, $\supseteq$ - Superset, $A \backslash B$ - **relative complement** (takes all in elements in A that are NOT in B), $|A|$ - Cardinality, P(A) - Power set, $\times$ - Cartesian product. Other notation you should know!
- Cartesian product - the set of all **ordered** pairs formed from two or more sets. It's like creating an array from the matrix that would be generated from vector multiplication. Generally, $\prod_i X_i = X_1 \times X_2 \times ... = (x_1, x_2, ...) | x_i \in X_i$
- Power set - the set of all **subsets** (not proper subsets) of a given set. Recall $2^|A| = |P(A)|$. Also don't forget the empty set!
- Note, subsets require curly brackets! Elements do not.
- Russel's paradox - $A = x | x \notin A$

- Axiom of Choice - A pretty straightforward law, which is clearly only of use to serious mathematicians when they need to prove something. It states that the Cartesian product of **non-empty** sets is **non-empty** ($\prod_i X_i \neq \emptyset \forall X_i \neq \emptyset$) (no shit sherlock). I guess it is referred to as the axiom of choice because it means you always have a choice to chose from elements (or tuples) in a set if that set is the Cartesian product of other NON-EMPTY sets.

1.3 Construction of numbers

- **Peano construction** $\rightarrow$, $0 = 0, 1 = S(0), 2 = S(S(0)), 3 = S(S(S(0)))$...
  - Peano postulates: 1) Zero is a natural number, 2) For natural numbers **equality** is reflective, symmetric and transitive, 3) $\mathbb{N}$ is closed under equality, 4) There is an 'successor' operator S such that $S(x)$ is a $\mathbb{N}$ for every x that is a natural number, 5) For two $\mathbb{N}$ x, y, $x = y$ iff $S(x) = S(y)$, 6) 0 is not a successor of any natural number, 7) Axiom of Induction - if K is a set such that it contains 0 and for every natural number n, n being in K which implies $S(n)$ is in K, then K contains all $\mathbb{N}$ (means we can use induction as proof method).
  - Definition of Addition: $a+b \equiv P(a,b)$ where $P : \mathbb{N}\times\mathbb{N} \rightarrow \mathbb{N}$, such that $\boxed{1)\ a+0=a \text{ and } 2)\ a+S(b)=S(a+b)}$. E.g. $1+1 = 1+S(0)(rule2) = S(1+0) = S(1)(rule1) = 2$
  - Definition of Multiplication $a \cdot b \equiv ab \equiv M(a,b)$, where $M : \mathbb{N}\times\mathbb{N} \rightarrow \mathbb{N}$, such that $\boxed{1)\ a\cdot 0 = 0,\ 2)\ a \cdot S(b) = a + a\cdot b}$
- **Von Neumann Construction** - $0 = \emptyset$ and $n+1 = n \cup n \rightarrow 0 = \emptyset, 1 = 0\cup 0 = 0, 2 = 1\cup 1 = 0\cup 1 = 0, 1, 3 = 2\cup 2 = 0, 1\cup 2 = 0, 1, 2$

1.4 Groups

Recall, this is means of understanding groups/clusters of numbers that behave well together and can be used for proofs.

1.4.1 Definitions

- A $\boxed{\underline{\text{group}}\ (\mathbf{G}, \cdot)}$, is a set **G** and a binary operation f(x,y) or $x \cdot y$, which looks something like this: $(\{1,2,3,4\}, \times)$. It must satisfy the following four conditions:
  - **Closure** := $x, y \in \mathbf{G}$ then $x \cdot y \in \mathbf{G}$ $(G \cdot G \rightarrow G)$
  - **Associativity** := $\forall x, y, z \in \mathbf{G}$ then $x \cdot (y \cdot z) = (x \cdot y) \cdot z$. Note: Binary operations multiplication and addition are always associative.
  - There must be an **identity** $1 \in G$ or $e \in G$, such that $x \cdot 1 = 1 \cdot x = x$
  - Each element in **G** has an **inverse** element that is also in **G** := $x \cdot x^{-1} = 1$.
- An Abelian Group is a group with **commutativity**, i.e. $\forall a, b \in G, a \cdot b = b \cdot a$
- Hierarchy: Assuming all have closure: Magma or Groupoid (always), Semigroup (if associative), Monoid (if associative and has identity element), Group and Abelian Group

1.4.2 Fermat's little theorem

- For any prime p and any integer a, $a^p - a$ is an integer multiple of p. This can be re-written as $a^p - a = np \rightarrow a^p - a = 0 \text{ (mod p)} \rightarrow a^p = \text{a (mod p)} \rightarrow \mathbf{a^{p-1} \bmod p = 1 \bmod p}$
- Recall (e.g.) *mod 5* simply means if a number is greater than 5 we keep subtracting 5 until it is smaller than 5.

1.5 Lagrange and Fermat

- (For my own clarity, let G represent the set in the group and $\mathbf{G} = (G, \cdot)$ the group, i.e. the set and the operation.)
- Subgroup - Given a group $\mathbf{G} = (G, \cdot)$, then $\mathbf{H}$ is a subGROUP if $H \subseteq G$ AND it is still a **group** under operation $\cdot$
- Proper subgroup - $\mathbf{H}$ is a proper subgroup if additionally $H \subset G$, which can be denoted by $H < G$
- Cosets - A set that is not a subset but is related to a subset by a simply operation (transformed/shifted subset). **Left Coset** - Given subgroup $H \subseteq G$ and $a \in G$ the set $K = aH = a \cdot H = \{a \cdot h | h \in H\}$. **Right Coset** - $K = Ha = H \cdot a = \{h \cdot a | h \in H\}$. Note for Abelian groups left and right cosets are the same. E.g let $a \cdot H$ mean that you add a with each of the elements in the set H - so $K = 1 \cdot \{0, 3\} = \{1, 4\}$
  - Observations: Let $H = \{0, 2, 4, 6\}$ and $G = \{0, 1, 2, 3, 4, 5, 6, 7\}$, then there exists for any $a \in G$ a $b \in G$ such that $a \in bH$, e.g $1 \in G$, $5 \in G$ and $1 \in 5 \cdot \{0, 4\} = \{5, 1\}$.
  - For two finite cosets aH and bH, $|aH| = |bH|$
  - If two cosets contain the same elements then they are the same coset: $aH \cap bH = \emptyset$ or $aH = bH$
  - Essentially, all this means all the elements of G get divided up equally amongst the cosets. In **Lagrange's** words "Given a finite group $\mathbf{G}$ and a subgroup $H \subseteq G$, $|H|$ divides $|G|$" (**Lagrange Theorem**)
    - E.g. If $|G| = 323 = 17 \times 19$ and therefore has prime factors $1, 17, 19, 323$ then there must be four subgroups - note that $|H| = 1$ is the 'trivial subgroup' (made of the identity element) and $|H| = |G| = 323$,i.e. itself $G = H$. Note, the statement is an 'if then' statement - just because the cardinality is divisible into prime factors, does not mean it has subgroups.
- Cyclic subgroups (Use this as a trick to generate subgroups) - given a finite group $\mathbf{G}$ and an element $a \in G$ the cyclic subgroup is defined as $< a >= \{e, a, a^2, a^3, ...\}$, where $a^2 = a \cdot a$ and e is the identity. Furthermore, $r \in \mathbb{N}, r > 0$, must exist such that $a^r = e$. r is therefore the "period" of the set or simply the cardinality $| < a > | = r$. Also, $< a >$ is **always** a subgroup of $\mathbf{G}$.
  - Corollary to the Lagrange theorem: "Given a finite group G and $a \in G$, the order r ($| < a > |$) of a divides $|G|$. Obvious really because $< a >$ is, like H, a subgroup of G.
- Proof of Fermat's Little Theorem Not happy with notes-proof because it only works for $a < p - 1$ so go over it after all other notes are done

1.6 Vector Space

1.6.1 Ring

- A ring is an algebraic structure that satisfies the following conditions:
  - $(A, \oplus)$ is an abelian group
  - $(A, \bullet)$ is a semi-group.
  - Operation $\bullet$ distributes over $\oplus$.
  - (Commutative rings have the operation $\bullet$ being commutative as well)
  - (A ring with unity has an identity for operator $\bullet$)

1.6.2 Field

- A field is an algebraic structure that follows the following properties:
  - $(A, \oplus)$ is an Abelian group
  - $(A \backslash \{0\}), \bullet)$ is an Abelian group
  - $\bullet$ distributes over $\oplus$

1.6.3 Vectorspace/Vectorfield

- Definition of a vectorspace V over the scalar field F for vector addition $V+V \rightarrow V$ and scalar multiplication $F \cdot V \rightarrow V$:
  - V with $\oplus$ is an Abelian group
  - $\cdot$ is associative - $(a \cdot b) \cdot v = a \cdot (b \cdot v), \forall a, b \in F$ and $v \in V$
  - Distributivity of scalar sums - $(a + b) \cdot v = (a \cdot v) + (b \cdot v), \forall a, b, \in F$ and $v \in V$.
  - Distributivity of vector sums $a \cdot (v + w) = (a \cdot v) + (a \cdot w), \forall a \in F$ and $v, w \in V$
  - Multiplicative identity - There exists $1/e \in F$ such that $1/e \cdot v = v, \forall v \in V$

1.6.4 Basis

- Linear Dependence - A set of vectors is considered linearly dependent one of the vectors can be expressed as a weighted sum of the others. If none of the vectors can be expressed by other vectors then the create a set of independent vectors (like $\{i, j, k\}$ - the unit vectors of a coordinate system).
- All elements of a Basis are linearly independent and any vector not in the basis can be expressed as a linear combination of basis vectors.
  - Examples: 3D coordinate system (i,j,k), Taylor expansion $(1, x, x^2)$, Fourier series $(cos(x), sin(x)$ (the individual wave functions that make up the larger more complicated one), PCA (principle component analysis - eigenvectors are the basis) - or for solving matrix differential equations.

1.6.5 Matroids

- A matroid is a pair (E, I) such that 1) E is a set of elements (Ground set) and 2) I is a set of sets of elements of E (independent set) and
  - Foundation Property - $\emptyset \in I$ The empty element is independent.
  - Hereditary Property - For $X' \subset X \subset E$ if $X \in I$ then $X' \in I$. Every subset of an independent set is independent.
  - Augmentation Property - For $X, Y \subset I$ and $|X| > |Y|$, there is an $x \in X$ such that $x \cup Y \in I$. If there are two independent sets of different sizes then one can always take an element from the bigger set and add it to the smaller set without destroying independence.

Now, all vectorspaces are matroids, but not all matroids are vectorspaces, but that's fine - Matroids are much better because there are guaranteed basis (therefore the same goes for vectorspaces) and one can find the OPTIMAL basis using a **greedy algorithm**.

- Greedy algorithms, like Kruskal's algorithm, build an optimal solution by choosing the best **local** options at a time. Generally this can lead to bad solutions, but if the system is a matriod structure, then it will lead to the best solution (apparently). So the key is to find out if your system is a matroid and then you can apply greedy algorithms ;). (Spanning trees are matroids)

1.7 Incompleteness

- **Hilbert's Aim** - To prove that mathematics is complete and consistent, i.e. that every statement can be proved true or untrue (complete), but not both (consistent, i.e. no paradoxes).

1.7.1 Cantor's diagonal argument

- Can be used to show that the Real numbers and the **cardinality of the powerset of natural numbers** are uncountably infinite, while the cardinality of natural numbers are countably infinite. (I wonder if counting in a 'different dimension changes things'). This is shown by creating a set (or number) that contains elements on the diagonal of the list of numbers and can therefore not be in the set of countable numbers.

1.7.2 First Incompleteness Theorem

- Das Entscheidungsproblem - show that any mathematical statement can be shown in finitely many steps that the statement is true or false.
- Through the following steps (which I am not going to go into until I see a question on it) were used by Gödel: 1) Arithmetization of Syntax, 2) Arithmetization of Provability, 3) Diagonalisation. Only to realise that there are some statements we cannot prove (incompleteness) although they are true and others that we can prove although they are wrong (paradoxical)

1.7.3 Second Incompleteness Theorem

- Gödel proved that no formal system can prove its own consistency. And we can never prove that maths is free of paradoxes. "This statement is unprovable".

2 Computability - Trevor - 8 lectures

2.1 Finite Automata

- Automaton - a machine that automatically follows a sequence of instruction. Simple automata can recognise simple expressions and properties. More complex automata can evaluate arbitrary expressions and compile computer programmes. Automata theory reveals limitations on computation and efficiency.
- A fine automaton is a model of a simple computational process with limited inputs and states and is typically modelled as a state graph. **Deterministic finite automaton** is a finite automaton without choice, i.e. it has determined outcomes. A **DFA** is represented as a 5-tuple, $(Q, \sum, \delta, q_0, F)$, each with the following properties/definitions:
  - Q - a **finite set** of states, e.g. $\{q_0, q_1, q_2\}$ or $\{open, closed\}$
  - $\sum$ is a finite set of symbols, e.g. $\{0, 1\}$ or $\{a, b, c, d\}$
  - $\delta : Q \times \sum \to Q$ is the transition function, i.e. it relates states with the symbols and says that a state acted upon by a symbol like $\{on, off\}$ will bring you back to the same or a different state, but a state nonetheless.
  - q_0 is the start state, so $q_0 \in Q$
  - F is a set of final states, so $F \subset Q$
- **DFA's are used for lexical analysis (processing strings) and finding patterns.**
- A **string** is a sequence of concatenated symbols from the alphabet $\sum$.
- $\sum^*$ (Sigma with a 'kleene star') is an infinite set containing all the possible strings in the alphabet including the empty string ϵ, e.g. $\sum^* = \{\epsilon, 0, 1, 01, 10, 00, 11, 000, ...\}$. A **Language**, L, therefore is a subset of the strings over an alphabet, $L \subseteq \sum^*$, i.e. it is a certain collection of strings.
- Maths and notation
  - a, b, c, d are used to represent 'symbols' and u, v, w, x are used to represent 'strings' (i.e. collections of symbols).
  - The transition function can be used on strings as well as symbols, e.g. $\delta(q_0, a) = q_1$ and $\hat{\delta}(q_0, w) = \hat{\delta}(q_0, uv) = \hat{\delta}(q_1, v) = q_2$ if say $w = uv$ (use hat for strings).
  - The language $L(M)$ is the language that is accepted by DFA or accepting state M - which is known as **Regular Language**. E.g. if a language contains all combinations of words from the latin alphabet, but M only accepts German words, then $L(M)$ is the German language.
  - For any regular language there exists a unique DFA with minimum number of states
  - DFA's can be created from simpler DFA's by regular operations:

* Let $M_A = \{Q_A, \sigma, \delta_A, q_{A0}, F_A\}$ and $M_B = \{Q_B, \sigma, \delta_B, q_{B0}, F_B\}$ then $M = \{Q, \sigma, \delta, q_0, F\}$. Then the following holds: $Q = Q_A \times Q_B$ (cartesian product) which means $q = (q_A, q_B)$, $q_0 = (q_{A0}, q_{B0})$, (initial state holds both M_A and M_B's initial states), $\delta(q, a) = \delta((q_A, q_B), a) = (\delta(q_A, a), \delta(q_B, a))$ and for:
* Union: $L(M_1) \cup L(M_2)$
  - $F = (F_A \times Q_B) \cup (Q_A \times F_B)$
  - Proof showing that $w \in L \Rightarrow w \in (A \cup B)$.
* Concatenation: $L(M_1) \cdot L(M_2)$
  - $F = (F_A \times Q_B) \cap (Q_A \times F_B)$
* Kleene star: $L(M)^*$
  - I think it just means you can extend the DFA by adding more states...
* Complement $\neg L(M)$
  - Accepts all strings that are not in language $L(M)$

2.1.1 Non-deterministic finite automaton

- The same as a DFA except $\delta : Q \times \sum \rightarrow 2^Q$, i.e the transition function returns a set of states (power set), rather than a single one, like the DFA.
- Start in q_0 then for each symbol in w determine all possible states reachable from the current set of states and when all symbols in w have been used, w is accepted iff at leat one current state is final. Essentially this means you need to test a bunch of different possible options for a string and if one of them works then it is considered 'accepted' by the NFA.
  - NFA's are not practical but important theoretically and can be created from **regular expression** - which is often used in text searches.
    * | indicates alternatives
    * * indicates zero or more occurrences
    * () is used for grouping
      - e.g. (0—1)*01 could be 000001 or 0101010101 etc...
- Going from NFA to DFA - the algorithm is as follows, but it is very intuitive and best to do with examples:
  - Let $Q_D = 2^{Q_N}$, i.e. the set of states of deterministic Q is the power set of non-deterministic Q.
  - $F_D = \{S | S \subseteq Q_N \wedge S \cap F_N \neq \emptyset\}$ and finally define $\delta_D(S, a) = \cup_{q \in S} \delta_N(q, a)$, where $a \in \sum$ and $S \subseteq Q_N$. Trick is two write the transition table and only look at the states that are reachable from the powerset of states.
  - Induction Proof that the same language is accepted when you go from NFA to DFA is very nice and simple and worth reviewing before exam.
- **Limitations:** binary strings of the form $0^n 1^n$ for any n are not regular language so they cannot be accepted by DFA's. learn proof
- **Mearly** and **Moore** machines are examples of finite automaton machines that can be used for all sorts of different applications.

2.2 Turing Machines

- An 'infinite' tape acts as a memory. A read/write head can read and write over the current cell symbol and be moved left and right (or the tape can be moved left and right - same thing). **A Turing machine** is simply a tape connected to a DFA. It is a 6-tuple: $TM = \{Q, \sum, \Gamma, \delta, q_0, F\}$. Only difference to DFA is:
  - Γ is a finite set of symbols (tape alphabet) - it includes empty cell and starting cell ($\triangleright$) $\sum \subset \Gamma$
  - $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times L, R$, i.e. the transition function takes a tape alphabet (read) and a state and outputs a state and a new symbol (write) and moves the tape left or right.

- A Turing machine only stops/**halts** when:
  - It reaches a final state - computation succeeded
  - It reaches a final state where no transition is possible - computation failed.
- Feel free to use "MoveToTapeStart" to save writing extra states
- For multiple inputs add # to tape alphabet Γ.
- Note, to double a binary number just add zero to the end. For something like $n \to 2n + 2$ just double then increment ($f \circ g(x) = f(g(x))$ - composition).
- What can Turing machines do? Evaluate functions, decision problems, accept/reject language strings, process language strings (e.g. copy a word) and enumerate language strings.
- Maths behind computation: A computation step - $C_i \vdash C_{i+1}$ is a single transition taking the machine from configuration C_i to C_{i+1}, where $C = uqv$, where q is the current tape position and uv is the tape content. So you have strings either side of the current state (first symbol in v is the current tape position). A computation is $C \vdash^* C_n$ (i.e. many transitions). Starting configuration therefore is $q_0 w$ and accepting/rejecting configuration is $uq_{accept/reject}v$, $w = uv$
- **Church-Turing thesis** "Any effective procedure can be encoded as a Turing machine". (unprovable hence thesis, because 'effective procedures are an informally defined set, will Turing machines are well defined).
  - Effective procedure means computations are finite, conclusive (i.e. they terminate) and deterministic.
  - A function is **Turing-computable** if all its input-output pairs are produced by a Turing machine, i.e. it can be encoded by a Turing machine.
- Turing machines can be extended infinitely in both directions of the tape, have multiple tapes and non-deterministic automaton, but all extensions are **equivalent** to a standard Turing machine. **Equivalence** means they can compute the same functions, but does not mean they can do it with equal efficiency.
- A **Universal Turing Machine** can read an encoding of another Turing machine and its input to produce the same output - essentially it is a programmable computer.
- **The Halting Problem** - Halting is not computable. No Turing machine can be made to decide whether or not an arbitrary Turing machine will eventually halt when presented with an arbitrary input w. By the Church-Turing thesis halting is not effectively computable.
- Other models of computation include "Cellular automata"

2.3 Computational Complexity

- Definitions:
  - **Turing complete** - A device or programme that can act as a Turing machine
  - A language is **Turing-recognisable** if strings in language L are accepted by a Turing Machine and are rejected or loop if a string is not in L
  - **Turing-decidable** means L is accepted or rejected - no looping. Therefore every Turing decidable language is Turing recognisable but not the other way around.
  - A **decider** is a TM that halts on all input - i.e. it is a TM that works for all inputs? Universal TM?
  - **Time and Space Complexity**
    - The time complexity of an algorithm (TM) is a function $f : N \to N$ where $f(n)$ represents the **maximum** number of **transitions** that occur in processing an input of size n (tape cells). It is represented using "big-O" notation. Examples:

- (**TIME(t(n))** defines a class that is the set of languages decided by $O(t(n))$: Linear - Execution time: $O(n)$ (finding smallest element in list), linearithmic - $O(nlogn)$ (quicksort), quadratic $O(n^2)$ (bubblesort), cubic $O(n^3)$ (naive matrix multiplication), polynomial $O(n^k)$ (checking if prime), exponential $O(2^n)$ (shortest path between vertices), factorial $O(n!)$ (all paths between vertices)
- If a problem is TIME(t(n)) for multi-tape then it is usually TIME($t^2(n)$).

* The space complexity of an algorithm (TM) is a function $f : N \to N$ where $f(n)$ represents the **maximum** number of tape cells that are used in processing an input of size n (tape cells).

2.3.1 Polynomial time

.

- Programmes that run on polynomial time are considered 'feasible'. Exponential (Exponentiation), Factorial, Tetration ($a^{a^{a^a}}$)) are considered infeasible. The **class P** is the set of all languages (problems) that are decidable in polynomial time of a deterministic single-tape Turing machine. $P = \bigcup_k TIME(n^k)$
- **Class NP** (Non-deterministic Polynomial time) is the set of all languages (problems) that are decidable in polynomial time of a non-deterministic single-tape Turing machine. $P = \bigcup_k NTIME(n^k)$ (non-deterministic time). If a problem can be **checked** in polynomial time it is considered NP, if it can be solved in polynomial time it is considered P.
- $P = NP$? Asks whether a quickly checkable problem (NP (checked in polynomial time)) can be solved in polynomial time too (P).
- A problem is $NP - complete$ if any other problem in NP can be reduced to it.

3 Applications - Trevor - 3 lectures

3.1 Cryptography

- Symmetric Key System - Both the sender and recipient share the same secret key.
- Public key system - A combination of a public key, which can be shared with anybody and private keys which only the owner knows.

3.1.1 Public key cryptography

- RSA Algorithm: 1) Choose two primes p, q and let $N = pq$ 2) Chose e such that $1 < e < \phi(N)$, where $\phi(N) = (p-1)(q-1)$, and e is coprime (doesn't share any common factors) with $N, \phi(N)$. 3) Choose d such that $ed \ mod \ \phi(N) = 1 \ mod \ \phi(N)$. Then encryption key is (e, N) and decryption key is (d, N). Encryption is $c = m^e \ mod \ N$ and decryption is $c^d = m \ mod \ n$.
- It works because of the Chinese remainder theorem and Fermat's little theorem. Proof is to show that $c^d = m^e d \ mod \ N = m \ mod \ N$.
- Each step requires polynomial time. So the total process is in polynomial time.

4 Game Theory - Thilo - 5 lectures

4.1 Intro to Game Theory

4.1.1 Definitions

- **Decision Problem** - You choose from a set of actions $S = \{A\}$ which have outcomes. There is a preference relation regarding the outcomes $\{a, b, c\}$. Decision problems can be represented as a decision tree.

- **Preference relation** - *Complete* iff any two outcomes can be ranked by the preference relation. It is *transitive* iff for outcomes $\{a, b, c\}$, 'if $a \succ b$ and $b \succ c$ then $a \succ c$' holds.
- **Payoff function** (Only valid if preference relation is complete and transitive) Let X be the space of outcomes, then $P : X \to R$ such that for any pair $x, y \in X, p(x) > p(y)$ if $x \succ y$ (just gives a numerical value to a choice).
- **Rational Choice** - The player knows **all** actions, outcomes, relations between actions and outcomes, his preferences - the payoff function P. A rational player (homo economicus) chooses the action with the **highest** payoff.
- **Expected payoff** - The payoff a player is expected to get if he makes a certain choice. The expected payoff also assumes the greatest payoff, as this is what a rational player would choose.
- **Value of Information** - It's a numerical value given to the 'worth' extra information adds to the already provided options. $V = -P^* + \sum_i p_i P_i$, where P^* is the expected payoff without the new information (, p_i the probability of making action i, P_i the expected payoff P_i given the new information.
- **Risk Averse** - choosing less risky option if payoff is equal.
- **Risk Loving** - choosing the more risky option if payoff is equal.

4.1.2 Condorcet Paradox

- Three rational players with different preference relations, making majority vote choices behave irrationally.

4.1.3 Bernoulli's game and St. Petersburg paradox

- A pay-in of £1000 for getting £2^n in return if you get a head on the n-th flip of a coin, i.e. only tails before the n-th flip. The expected payoff is ∞, so paradoxically a sign for risk avoidance by people. Expected Payoff is: $P = -1000 + 0.5(1) + 0.5^2(2) + 0.5^3(4)... = -1000 + 0.5 + 0.5 + 0.5... = \infty$

4.2 Two-Player games

4.2.1 Definitions

- **Payoff matrix** - Finite, one-shot, two player games can be represented by actions and outcomes in a matrix.
- **Finite** - There are only finite many actions.
- **Symmetric** - Identical for both players
- **One Shot** - Players cannot respond to actions of opponents
- **Complete information** - Actions, Outcomes, and their relation are common knowledge.
- **Strategy** - A sequence of allowed actions.
- **Strategy profile** - A vector of strategies that assigns one strategy to every player.
- A strategy (profile) **pareto dominates** another strategy if it leaves *at least one player* a larger payoff and *no player with a lesser payoff*
- A strategy *profile* is **pareto optimal** if it is not pareto dominated by another strategy profile.
- **Social optimum** - The *combined* payoff of players is maximal.
- A strategy **strictly dominates** another strategy if it yields a better payoff across all *strategy profiles* (i.e. it has a higher expected payoff). A strategy **weakly dominates** another strategy if it yields at least the same payoff in all cases.

- **Nash equilibrium** - A strategy profile in which no player can achieve *identical* or *superior* outcome by changing their own strategy.
- A strategy profile in which every player can achieve a superior strategy by changing their own strategy is called a **weak Nash equilibrium**.

4.2.2 Cooperation games

- All one-shot, symmetric, two-player cooperation games have a payoff matrix of the following form:

	P2 C	P2 D
P1 C	Reward for cooperation	Sucker's payoff
P2 D	Temptation to defect	Punishment for mutual defection

- Prisoner's dilemma is the case where $T > R > P > S$

4.3 Iterated Games

- A strategy space contains all possible different strategies of an n-round game for one player. For two player games with two choices (e.g. Defect or Cooperative) have $2^{(2^{(n-1)})}$ choices given n rounds. It's not the same as playing n games sequentially, but rather you can have a strategy for one game that has n-rounds. The number of possible strategies is $\prod_{n=1}^{N} 2^{(2^{(n-1)})} = 2^{\sum 2^{n-1}} = 2^{2^{n-1}} = 0.5 \cdot 2^{(2^n)}$

4.3.1 Strategies with limited memory

- Examples - Always cooperate (AllC), Always Defect (AllD), Cooperate first then defect (CD), Cooperate first then do what the other player did (pTFT - positive tit for tat), Defect first, then do what the other player didn't do (nTFNT).
  - You can compare strategies using computer simulations in a payoff matrix. TFT normally does quite well.

4.3.2 Auctions

- English auction - auctioneer raises prices (competitive - snowdrift-game - $T > R > S > P$)
- Dutch auction - Auctioneer lowers prices. (Opportunity taking - minority game - $T > S > R > P$)
- Dollar Auction - proposed by Martin Shubik in 1950 as a model for arms races - A dollar is auctioned. Usually leads to overbidding (due to "invested-too-much-to-quit" trap)

4.3.3 Mixed and Pure strategies

- A strategy is mixed if it contains an element of randomness. They switch randomly between deterministic pure strategies.
- A strategy is pure if it is deterministic.
- Parrondo paradox - switching between losing strategies can lead to winning.
- Avoid second guessing spirals!

4.4 Continuous games

- **Payoff function** now is becomes a continuous function like $P(t) = t^2 - t$ for example.
- **Value of Information** (idea extrapolates to continuous domain) - $V = \int (P_{info}(T) - P_{noinfo}(T))p(T)dT$

4.4.1 Common good games

- Generally common good games, such as the 'tradegy of commons', are created so that it is better for the common/group to work together, but better for the individual to work alone.
- The payoff function now becomes a two variable function, could be more if needed - $P(x, y) = ...$
  - Again Pareto dominance - a strategy profile (x', y') Pareto dominates another strategy profile (x, y) if **all** players receive a better payoff - applies. So does Nash equilibrium, i.e. the strategy profile, in which each player is at their own best strategy (i.e. the one that gives them the highest expected payoff)
  - **Hotelling's law** - Similar business, will always end up close together and selling similar products.
    - Beach vendor scenario - in which two beach vendors will end up at the centre (nash equilibrium) is an example of this
  - **Zero-sum game** - A player can only increase its payoff if the other player's payoff decreases by the same amount.

4.5 Evolutionary games

This is essentially game theory applied to biology and in particular to evolution. It allows us to think about evolutionary stability (will a species continue to thrive?) and instability (will it die out?), adaptability (can the species adapt to changes?), mutations (will species with different 'strategies' take over the population, i.e. will the species evolve?), etc...

4.5.1 Classification of Outcomes

- **Weak selection** - The strategies in the population only change slowly and locally due to mutations.
- **Mutant** - An individual/agent of a population that follows a different strategy to the rest of the population.
- **Adaptive dynamics** - A means of formulating macroscopic theories for microscopic games.
- **Selection exponent or Fitness** - the pay-off function of a mutant using strategy $m = r + \delta$, where r is the mean strategy of the population. (So the mutant strategy is close (gaussian distribution) to the population's strategy, yet different.) - $S_r(m) = P(m, r)$. To decide which mutants will survive we consider the fitness function with different values of m.
  - **Selection Gradient** - The derivative $S'(r) = \frac{\partial}{\partial m} S_r(m)|_{m=r}$. If $S'_r(r) > 0$ then mutants with $m > r$ will thrive, while mutants with $m < r$ will vanish - in this case r increases with time. If $S'_r(r) < 0$ then r decreases with time.
  - **Evolutionary drift** - when the resident strategy r changes in time according to $\frac{d}{dt} r \sim S'_r(r)$
  - **Evolutionary Singular Strategies (ESS)** - Resident strategies that have the following property: $S'_r(r) = 0|_{r=r^*}$
    - Stability requires that $\frac{\partial}{\partial r} S'_r(r)|_{r=r^*} < 0$ (second derivative) - this is referred to as **Convergence stable strategy (CSS)**.
      - If $\frac{\partial}{\partial r} S'_r(r)|_{r=r^*} > 0$ then ESS is considered as a 'Evolutionary Repellor'
      - If $\frac{\partial}{\partial r} S'_r(r)|_{r=r^*} < 0$ and $S''_{r^*}(r^*) < 0$ then ESS is considered as a 'Stable Evolutionary Attractor'
      - If $\frac{\partial}{\partial r} S'_r(r)|_{r=r^*} < 0$ and $S''_{r^*}(r^*) > 0$ then ESS is considered as a 'Unstable Evolutionary Attractor'
  - Essentially - Mutants take over resident population if they have higher fitness, this leads to a 'drift' (change of population) proportional to the selection gradient and eventually we a CSS and the 'drift' stops

4.5.2 Red Queen effects

- **Evolution can lead to a fitness minimum** - since fitness maximum $S''_r(r)|_{r=r^*} \neq \frac{\partial}{\partial r} S'r(r)|_{r=r^*}$ convergence stability.
- Evolution can run in cycles
- You have to evolve as a fast as possible to stay in the same place.
- A selection gradient can lead to an unstable attractor/disruptive selection.