Administrative Stuff

- Evaluation:
  - 1 Midterm Exam
  - 1 Final Exam
  - 6-8 Homeworks
What is this course about? – Formal Languages

- An abstraction of the notion of a “problem”
- Problems are cast either as **Languages** (= sets of “Strings”)
  - "Solutions" determine if a given “string” is in the set or not
    - e.g., Is a given integer, $n$, prime?
- Or, as **transductions between languages**
  - “Solutions” transduce/transform the input string to an output string
    - e.g., What is $3+5$?
What is this course about? – Formal Languages

- So essentially all computational processes can be reduced to one of
  - Determining membership in a set (of strings)
  - Mapping between sets (of strings)
- We will formalize the concept of mechanical computation by
  - giving a precise definition of the term “algorithm”
  - characterizing problems that are or are not suitable for mechanical computation.
What is this course about? – Automata

- Automata (singular *Automaton*) are abstract mathematical devices that can
  - Determine membership in a set of strings
  - Transduce strings from one set to another
- They have all the aspects of a computer
  - input and output
  - memory
  - ability to make decisions
  - transform input to output
- Memory is crucial:
  - Finite Memory
  - Infinite Memory
    - Limited Access
    - Unlimited Access
We have different types of automata for different classes of languages.

They differ in:
- the amount of memory they have (finite vs infinite)
- what kind of access to the memory they allow.

Automata can behave **non-deterministically**
- A non-deterministic automaton can at any point, among possible next steps, pick one step and proceed.
- This gives the conceptual illusion of (infinitely) parallel computation for some classes of automata.
  - All branches of a computation proceed in parallel (sort of).
- More on this later.
What is this course about? – Complexity

- How much resource does a computation consume?
  - Time and Space

- What are the implications of nondeterminism for complexity?

- How can we classify problems into classes based on their resource use?
  - Are there problems with very unreasonable resource usage (Intractable problems)?
  - How can we characterize such problems?
    - P vs. NP, PSPACE, Log Space
What is this course about? – Computability

What is computational power?
- Automaton 1 tells Automaton 2
  “Tell me what kinds of problems you can solve and I will tell you how powerful you are? “

What does computational power depend on? (it turns out, not “speed”)

What does it mean for a problem to be computable?

Are there any uncomputable functions or unsolvable problems?
- What does this mean?
- Why do we care?
Applications/Relevance

- Pattern matching
  - Perl Hacking
  - Bioinformatics
  - Lexical analysis

- Design and Verification
  - Hardware
  - Software
  - Communication Protocols

- Parsing Languages
  - Compiler construction
  - XML Analysis
  - Natural language processing, Machine Translation

- Algorithm design and analysis
A decision problem is a function with a YES/NO output.

We need to specify:
- the set $A$ of possible inputs (usually $A$ is infinite)
- the subset $B \subseteq A$ of YES instances (usually $B$ is also infinite)

The subset $B$ should have a finite description!
Decision Problems – Examples

- **A: integers**
  - is_even?(x)
  - is_prime?(x)

- **A: integers \times integers**
  - is_relatively_prime?(x,y)
Decision Problems – Examples

- $A$: set of all pairs $(G, t)$
  - $G$ is a finite set of triples of the sort $(i, j, w)$,
  - $i$ and $j$ are integers and $w$ is real
  - The finite set encodes the edges of a weighted directed graph $G$.
  - $A = \{\ldots (\{\ldots, (3, 4, 5.6), \ldots \}, 8.0), \ldots \}$

- Each pair in $A$, $(G, t)$, represents a graph $G$ and a threshold $t$
- Does $G$ have a path that goes through all nodes once with total weight < $t$?
  - Travelling Salesperson Problem

- $A$ is the set of all TSP instances.
Sets can be
- Finite
- Infinite
  - *Countably Infinite*: can be put in one-to-one correspondence with natural numbers (e.g., rational numbers, integers)
  - *Uncountably Infinite*: can NOT be put in one-to-one correspondence with natural numbers (e.g., real numbers)
Encoding Sets

- In real life, we use many different types of data: integers, reals, vectors, complex numbers, graphs, programs (your program is somebody else's data).
- These can all be encoded as strings
- So we will have only one data type: strings
Strings

- An **alphabet** is any **finite** set of distinct symbols
  - \{0, 1\}, \{0,1,2,...,9\}, \{a,b,c\}
  - We denote a generic alphabet by $\Sigma$

- A **string** is any **finite-length sequence** of elements of $\Sigma$.

- e.g., if $\Sigma = \{a, b\}$ then $a, aba, aaaa, \ldots,$.
  - $abababbaab$ are some strings over the alphabet $\Sigma$
Strings

- The **length** of a string \( \omega \) is the number of symbols in \( \omega \). We denote it by \( |\omega| \). \( |aba| = 3 \).
- The symbol \( \epsilon \) denotes a special string called the **empty string**
  - \( \epsilon \) has length 0
- **String concatenation**
  - If \( \omega = a_1, \ldots, a_n \) and \( \nu = b_1, \ldots, b_m \) then \( \omega \cdot \nu \) (or \( \omega \nu \))
    \[ = a_1, \ldots, a_n b_1, \ldots, b_m \]
  - Concatenation is associative with \( \epsilon \) as the identity element.
- If \( a \in \Sigma \), we use \( a^n \) to denote a string of \( n \) \( a \)'s concatenated
  - \( \Sigma = \{0, 1\} \), \( 0^5 = 00000 \)
  - \( a^0 =_{\text{def}} \epsilon \)
  - \( a^{n+1} =_{\text{def}} a^n a \)
Strings

- The reverse of a string $\omega$ is denoted by $\omega^R$.
  - $\omega^R = a_n, \ldots, a_1$

- A substring $y$ of a string $\omega$ is a string such that $\omega = xyz$ with $|x|, |y|, |z| \geq 0$ and $|x| + |y| + |z| = |\omega|$

- If $\omega = xy$ with $|x|, |y| \geq 0$ and $|x| + |y| = |\omega|$, then $x$ is prefix of $\omega$ and $y$ is a suffix of $\omega$.
  - For $\omega = abaab$,
    - $\epsilon, a, aba,$ and $abaab$ are some prefixes
    - $\epsilon, abaab, aab,$ and $baab$ are some suffixes.
Strings

- The set of all possible strings over $\Sigma$ is denoted by $\Sigma^*$.
- We define $\Sigma^0 = \{\epsilon\}$ and $\Sigma^n = \Sigma^{n-1} \cdot \Sigma$
  - with some abuse of the concatenation notation applying to sets of strings now
- So $\Sigma^n = \{\omega | \omega = xy \text{ and } x \in \Sigma^{n-1} \text{ and } y \in \Sigma\}$
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \Sigma^n \cup \ldots = \bigcup_{0}^{\infty} \Sigma^i$
  - Alternatively, $\Sigma^* = \{x_1, \ldots, x_n | n \geq 0 \text{ and } x_i \in \Sigma \text{ for all } i\}$
- $\Phi$ denotes the empty set of strings $\Phi = \{\}$,
  - but $\Phi^* = \{\epsilon\}$
Strings

- $\Sigma^*$ is a countably infinite set of finite length strings.
- If $x$ is a string, we write $x^n$ for the string obtained by concatenating $n$ copies of $x$.
  - $(aab)^3 = aabaabaab$
  - $(aab)^0 = \epsilon$
Languages

- A language $L$ over $\Sigma$ is any subset of $\Sigma^*$

- $L$ can be finite or (countably) infinite
Some Languages

- $L = \Sigma^* -$ The mother of all languages!
- $L = \{ a, ab, aab \} -$ A fine finite language.
  - Description by enumeration
- $L = \{ a^n b^n : n \geq 0 \} = \{ \epsilon, ab, aabb, aaabbb, \ldots \}$
- $L = \{ \omega | n_a(\omega) \text{ is even} \}$
  - $n_x(\omega)$ denotes the number of occurrences of $x$ in $\omega$
  - all strings with even number of $a$’s.
- $L = \{ \omega | \omega = \omega^R \}$
  - All strings which are the same as their reverses – palindromes.
- $L = \{ \omega | \omega = xx \}$
  - All strings formed by duplicating some string once.
- $L = \{ \omega | \omega \text{ is a syntactically correct Java program} \}$
Languages

- Since languages are sets, all usual set operations such as intersection and union, etc. are defined.

- Complementation is defined with respect to the universe $\Sigma^*$: $\overline{L} = \Sigma^* - L$
Languages

- If \( L, L_1 \) and \( L_2 \) are languages:
  - \( L_1 \cdot L_2 = \{ xy | x \in L_1 \text{ and } y \in L_2 \} \)
  - \( L^0 = \{ \epsilon \} \) and \( L^n = L^{n-1} \cdot L \)
  - \( L^* = \bigcup_{0}^{\infty} L^i \)
  - \( L^+ = \bigcup_{1}^{\infty} L^i = L^* - \{ \epsilon \} \)
Sets of Languages

- The power set of $\Sigma^*$, the set of all its subsets, is denoted as $2^{\Sigma^*}$. 

![Diagram showing the relationship between finite sets $\Sigma$, infinite sets $\Sigma^*$, and the power set $2^{\Sigma^*}$, which represents the set of languages/family/class of languages.](image)
Describing Languages

- Interesting languages are infinite
- We need **finite descriptions** of infinite sets
  - \( L = \{a^n b^n : n \geq 0\} \) is fine but not terribly useful!
- We need to be able to use these descriptions in mechanizable procedures