Mathematical Preliminaries

A set is a collection of elements. Those elements can be anything, including other sets.

Sets can be described by listing their elements, such as $\{a, b, c, \ldots\}$.

The empty set is denoted $\emptyset$, or $\{\,\}$.

A set A is *finite* if it has a finite number of elements, that is, if there is a natural number $n \in \mathbb{N}$ such that A has n elements. If no such n exists, then A is *infinite*.

The main relation on sets is *set membership*, written $a \in A$: a is an element of set A.

Sets are *extensional*: two sets are equal if they have exactly the same elements.

Another relation on sets is *set inclusion*, written $A \subseteq B$: A is a subset of B, meaning that every element of A is an element of B).

Some properties of $\subseteq$:

> $\emptyset \subseteq A$ for every A
>
> $A \subseteq A$ for every A
>
> If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$
>
> $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

If P is a property, then $\{x \mid P(x)\}$ is the set of all elements satisfying the property. (Technically speaking, there are some restrictions on what makes up an acceptable property in this context — but they won't impact us.)

Common operations on sets:

> $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
>
> $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
>
> $\overline{A} = \{x \in \mathcal{U} \mid x \notin A\}$ where $\mathcal{U}$ is a universe of elements with $A \subseteq \mathcal{U}$ — the definition of $\overline{}$ therefore depends on the universe under consideration, which will often be clear from context
>
> $A \backslash B = \{x \mid x \in A \text{ but } y \notin B\}$

$A \times B = \{\langle x, y \rangle \mid x \in A, y \in B\}$, the set of all pairs of elements from A and B. This generalizes in the obvious way to products $A_1 \times A_2 \times \cdots \times A_k$.

A function $f : A \longrightarrow B$ associates (or maps) every element of A to an element of B. Set A is the domain of the function, and B is the codomain. The image of A under f is the subset of B defined by $\{b \in B \mid f(a) = b \text{ for some } a \in A\}$.

If $f : A \longrightarrow B$ and $g : B \longrightarrow C$, then the *composition* $g \circ f : A \longrightarrow C$ defined by $(g \circ f)(x) = g(f(x))$.

A function $f : A \longrightarrow B$ is *one-to-one* if it maps distinct elements of A into distinct elements of B (that is, if $a \neq b$, then $f(a) \neq f(b)$). A function $f : A \longrightarrow B$ is *onto* if every element of B is in the image of A under f (that is, if for every element $b \in B$ there is an element $a \in A$ with $f(a) = b$). A function $f : A \longrightarrow B$ is a *one-to-one correspondance* if it is both one-to-one and onto.

Decisions Problems

Intuitively, a computation is a way to "implement" a mathematical function $f : A \longrightarrow B$. A big part of the course is to build an understanding of what that intuition means.

Arbitrary functions between arbitrary sets A and B is too broad a class of functions to work with. Historically, researchers have looked at two classes of functions to study computation:

1. **Natural number functions** of the form

$$f : \mathbb{N} \times \cdots \times \mathbb{N} \longrightarrow \mathbb{N}$$

2. **Decision problems** of the form

$$d : A \longrightarrow \{1, 0\}$$

   where 1 can be interpreted as true and 0 as false. Decision problems are predicates on the domain A.

For the first half of this course, we will consider study computability for decision problems. Example of decision problems include determining if a graph is planar, or determining if two natural numbers are coprime. Restricting to decision problems looks like a limitation, in that many functions of interest we may be interested in computing are not in that form, such as addition, but we can capture addition using a decision problems by considering the decision problem that takes three numbers and determines if the sum of the first two is equal to the third.

One reason why decision problems form an interesting class of problems for which to study computability is that they can be "summarized" using simple sets. You can "summarize" a decision problem

$$d : A \longrightarrow \{1, 0\}$$

using the set

$$S = \{a \in A \mid d(a) = 1\}$$

Given such a set S, you can *reconstruct* d readily, by taking

$$d(a) = \begin{cases} 1 & \text{if } a \in S \\ 0 & \text{otherwise} \end{cases}$$

It is easy to check that you get the same decision problem back. Thus, you can study a decision problem by studying the set S that summarizes it.

We will make a further assumption, that the domain A for decision problems will be a set of *strings*.

Let Σ be a non-empty finite set we will call the *alphabet*. A *string over* Σ is a (possibly empty) finite sequence of elements of Σ, usually written $a_1 \cdots a_k$, where $a_i \in \Sigma$. For example, `aaccabc` is a string over alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. It is also a string over alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. We will use u, v, w to range over strings.

The length of $u = a_1 \dots a_k$, written $|u|$, is k.

The empty string is written ϵ. It has length 0.

The set of all strings over Σ is denoted Σ^*. Note that this is an infinite set. (Why?)

If $u = a_1 \dots a_k$ and $v = b_1 \cdots b_m$ are strings over Σ, then the *concatenation* uv is the string $a_1 \cdots a_k b_1 \cdots b_m$. Note that $\epsilon u = u\epsilon = u$ for every string u.

We define $u^0 = \epsilon$, $u^1 = u$, $u^2 = uu$, $u^3 = uuu$, etc.

We will transform the problem of of determining whether a decision problem is computable into the problem of determining whether a set of strings is computable. But of course, what we will mean w hen we say that a set of strings A is computable is that the corresponding decision problem constructed from A as above is computable. Why do we do this? Because sets of strings are much easier to work with: they are sets, and we can manipulate sets in many different ways.