4 Mathematical Foundations

4.1 Sets

You should have seen sets before. This is review.

Additional reading: Cummings Chapter 3 about sets and reasoning about sets.

Definition 4.1. A *set* is an unordered collection of things without repetitions. The things in set S are called the *members* of S.

Definition 4.2. A *set enumeration* is one way to describe a set, by writing the members of the set in braces, separated by commas. For example, {2, 5, 9} is a set of three integers.

4.1.1 Finite and Infinite Sets

It is possible to list the members of a *finite* set. But some sets, such as the set of all positive integers, have infinitely many members. Here are a few common infinite sets.

$\mathcal{N}$	$\{0, 1, 2, 3, \ldots\}$
$\mathcal{Z}$	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathcal{R}$	the set of all real numbers

Note. Mathematicians usually define $\mathcal{N}$ to be the set of all *positive* integers, and that is what Cummings does. Computer scientists usually define $\mathcal{N}$ to be the set of nonnegative integers because 0 is so important to computer programs. I will use the definition above, including 0.

4.1.2 Set Comprehensions

A *set comprehension* is a way to describe the set of all values that have a certain property. Notation

$$\{x \mid p(x)\}$$

stands for the set of all values x such that $p(x)$ is true and notation

$$\{f(x) \mid p(x)\}$$

stands for the set of all values $f(x)$ such that $p(x)$ is true. Notation

$$\{x \in S \mid p(x)\}$$

is shorthand for $\{x \mid x \in S \wedge p(x)\}$ Here are some examples.

Set	**Description**
$\{x \mid x \in \mathcal{R} \ \wedge \ x^2 - 2x + 1 = 0\}$	$\{-1, 1\}$
$\{x \in \mathcal{R} \mid x^2 - 2x + 1 = 0\}$	$\{-1, 1\}$
$\{x \mid x \text{ is an even positive integer}\}$	$\{2, 4, 6, \ldots\}$
$\{x^2 \mid x \text{ is an even positive integer}\}$	$\{4, 16, 36, \ldots\}$

4.1.3 Set Notation and Operations

Table 4.1 defines notation for sets.

Note. Mathematicians commonly use operator $\setminus$ to mean set difference, and that is what Cummings does. That is, Cummings defines $S \setminus T$ to be the set of all members of S that are not members of T. Computer scientists usually write $S - T$ for set difference.

4.1.4 Identities for Sets

Table 4.2 lists some identities that are easy to establish.

Table 4.1	
Notation	**Meaning**
$\|S\|$	The *cardinality* (size) of S, when S is a finite set.
$\{\}$	The empty set, which has no members
$x \in S$	True if x is a member of set S. For example, $2 \in \{1, 2, 3, 4\}$
$x \notin S$	$\neg(x \in S)$
$S \cup T$	$\{x \mid x \in S \vee x \in T\}$. For example, $\{2, 5, 6\} \cup \{2, 3, 7\} = \{2, 3, 5, 6, 7\}$. This is called the *union* of sets S and T.
$S \cap T$	$\{x \mid x \in S \wedge x \in T\}$. For example, $\{2, 5, 6\} \cup \{2, 3, 7\} = \{2\}$. This is called the *intersection* of sets S and T.
$S - T$	$\{x \mid x \in S \wedge x \notin T\}$. For example, $\{2, 5, 6\} - \{2, 3, 7\} = \{5, 6\}$. This is called the *difference* of sets S and T.
$\overline{S}$	$U - S$, where U is the universe of discourse. This is called the *complement* of S.
$S \times T$	$\{(x, y) \mid x \in S \wedge y \in T\}$. For example, $\{2, 3\} \times \{5, 6\} = \{(2,5), (2,6), (3,5), (3,6)\}$. This is called the *cartesian product* of S and T.
$S \subseteq T$	This is true if $\forall x(x \in S \rightarrow x \in T)$. For example, $\{2, 4, 6\} \subseteq \{1, 2, 3, 4, 5, 6\}$. Notice that $\{2, 4, 6\} \subseteq \{2, 4, 6\}$. $S \subseteq T$ is read "S is a subset of T."
$S = T$	S and T are the same set if $S \subseteq T$ and $T \subseteq S$. That is, S and T have exactly the same members.

Table 4.2
Some Set Identities
$A \cup \{\} = A$
$A \cap \{\} = \{\}$
$\overline{\overline{A}} = A$
$A \cup B = B \cup A$
$A \cap B = B \cap A$
$A \cup (B \cup C) = (A \cup B) \cup C$
$A \cap (B \cap C) = (A \cap B) \cap C$
$\overline{A \cup B} = \overline{A} \cap \overline{B}$
$\overline{A \cap B} = \overline{A} \cup \overline{B}$
$A - B = A \cap \overline{B}$.
$A \cup (A \cap B) = A$
$A \cap (A \cup B) = A$

4.1.5 Sets of Sets

The members of sets can be sets. For example, if $S = \{\{1, 2, 3\}, \{2, 4, 6\}\}$ then $|S| = 2$, since S has exactly two members, $\{1,\ 2,\ 3\}$ and $\{2,\ 4,\ 6\}$.

Do not confuse $\in$ with $\subseteq$. If $S = \{\{1, 2, 3\}, \{2, 4, 6\}\}$ then

$\{1, 2, 3\} \in S$

$\{1, 2, 3\} \not\subseteq S$

$3 \notin S$

Notice that $\{\} \neq \{\{\}\}$. $|\{\}| = 0$ but $|\{\{\}\}| = 1$ since $\{\{\}\}$ has one member, the empty set.

4.2 Alphabets and Strings

Definition 4.3. An *alphabet* is a finite, nonempty set whose members we call *symbols*.

We will usually want to use small alphabets such as $\{a,\ b\}$ or $\{a,\ b,\ c\}$, where symbols a, b and c stand for themselves (letters of an alphabet).

It is conventional to call an alphabet Σ (upper case Greek letter sigma, indicating *symbol*).

Definition 4.4. If Σ is an alphabet, then a *string over* Σ is a finite sequence members of Σ. (In a sequence, order matters and there can be repetitions.) A string can have length 0.

I will write strings in double-quotes. For example, if $\Sigma = \{a, b, c\}$ then "aab" and "$ccccc$" are two strings over Σ.

A fundamental operations on strings is *concatenation*, where $s \cdot t$ indicates s followed by t. For example, "abc" $\cdot$ "aba" = "$abcaba$". Just as the multiplication symbol is usually unwritten between numbers, we will usually omit the concatenation dot between strings and write st to mean $s \cdot t$.

We will allow concatenation to work with symbols as well as strings. For example, "aab" $\cdot\ a$ = "$aaba$".

When the alphabet is understood or unimportant, we talk about a *string*, leaving the alphabet unstated.

Definition 4.5. If s is a string, then $|s|$ is the length of s (the number of characters in s). For example, $|"accb"| = 4$ and $|"b"| = 1$.

Definition 4.6. We write ε to mean the empty string, "", whose length is 0. (Symbol ε is a variant of Greek letter epsilon. Think of it as e for empty.)

4.2.1 Sets of Strings

Definition 4.7. A set of strings is called a *language*.

Definition 4.8. If Σ is an alphabet, then Σ^* is the set of all strings over Σ. For example, $\{a, b\}^* = \{\varepsilon, "a", "b", "aa", "ab", "ba", "bb", "aaa", \ldots\}$.

4.2.2 Natural Numbers as Strings

We will use strings as the inputs and outputs of algorithms or programs. But sometimes, we want the inputs and outputs to be integers. That is easy to manage: we write the integers in standard (base 10) notation as strings. For example, 25 is treated as string is "25".

4.3 Functions

You should have seen functions before. This is review.

Definition 4.9. If A and B are sets, then a *function with domain A and codomain B* associates exactly one value in set B with each value in set A. We write $f : A \rightarrow B$ to mean that f is a function with domain A and codomain B.

Definition 4.10. If $f : A \rightarrow B$ and $x \in A$, then notation $f(x)$ indicates the member of B that f associates with x. When $f(x) = y$, we say that f *maps* x to y.

For example, suppose that $f : \mathcal{N} \rightarrow \mathcal{N}$ is defined by $f(x) = x^2$. Then $f(3) = 9$ and $f(5) = 25$.

4.4 Computational Problems

We will look at two kinds of computational problems.

1. A *decision problem* is a problem where the input is a string (over a chosen *input alphabet*) and the output is either 1 (true) or 0 (false). We can also think of the output as yes or no.

   A decision problem can be expressed as a function or as a set of strings (a language). When S is a set of strings, we think of S as the decision problem:

   Input. String x over the input alphabet.

   Question. Is $x \in S$?

   Most of the problems that we look at will be decision problems.

2. A *functional problem* is a problem where the input is a string (over the *input alphabet*) and the output is a string (over the *output alphabet*).

4.5 Types

We will deal with several different types of things. It is essential that you know what type of thing each of your variables (or, in general, names) is.

Adjectives or other terms that we define can only be applied to certain types of things. For example, it makes sense to talk about the cardinality of a set, but not the cardinality of a number. The following is a list of some of the types of things that we will use.

Type	**Meaning**
boolean	A boolean value is either true or false. It might equally well be either 1 or 0, or either yes or no.
symbol	A symbol is a member of some alphabet.
string	A string is a (possibly empty) finite sequence of symbols
language	A language is a set of strings. We can think of a language as a decision problem.
function	Our functions will usually either take a string and yield a boolean value or will take a string and yield a string.
set of languages	A set of languages is called a *class*. We think of a language as a decision problem, and we will identify classes of decision problems that can be solved in particular ways.

prev **next**