Mathematics for Computer Science

revised Monday 18th May, 2015, 01:43

Eric Lehman
Google Inc.

F Thomson Leighton
Department of Mathematics
and the Computer Science and AI Laboratory,
Massachussetts Institute of Technology;
Akamai Technologies

Albert R Meyer
Department of Electrical Engineering and Computer Science
and the Computer Science and AI Laboratory,
Massachussetts Institute of Technology

Contents

4 Mathematical Data Types

We have assumed that you've already been introduced to the concepts of sets, sequences, and functions, and we've used them informally several times in previous sections. In this chapter, we'll now take a more careful look at these mathematical data types. We'll quickly review the basic definitions, add a few more such as "images" and "inverse images" that may not be familiar, and end the chapter with some methods for comparing the sizes of sets.

4.1 Sets

Informally, a *set* is a bunch of objects, which are called the *elements* of the set. The elements of a set can be just about anything: numbers, points in space, or even other sets. The conventional way to write down a set is to list the elements inside curly-braces. For example, here are some sets:

$$\begin{aligned} A &= \{\text{Alex}, \text{Tippy}, \text{Shells}, \text{Shadow}\} && \text{dead pets} \\ B &= \{\text{red}, \text{blue}, \text{yellow}\} && \text{primary colors} \\ C &= \{\{a, b\}, \{a, c\}, \{b, c\}\} && \text{a set of sets} \end{aligned}$$

This works fine for small finite sets. Other sets might be defined by indicating how to generate a list of them:

$$D ::= \{1, 2, 4, 8, 16, \ldots\} \qquad \text{the powers of 2}$$

The order of elements is not significant, so $\{x, y\}$ and $\{y, x\}$ are the same set written two different ways. Also, any object is, or is not, an element of a given set—there is no notion of an element appearing more than once in a set.[1] So, writing $\{x, x\}$ is just indicating the same thing twice: that x is in the set. In particular, $\{x, x\} = \{x\}$.

The expression $e \in S$ asserts that e is an element of set S. For example, $32 \in D$ and blue $\in B$, but Tailspin $\notin A$—yet.

Sets are simple, flexible, and everywhere. You'll find some set mentioned in nearly every section of this text.

[1] It's not hard to develop a notion of *multisets* in which elements can occur more than once, but multisets are not ordinary sets and are not covered in this text.

4.1.1 Some Popular Sets

Mathematicians have devised special symbols to represent some common sets.

symbol	**set**	**elements**
$\emptyset$	the empty set	none
$\mathbb{N}$	nonnegative integers	$\{0, 1, 2, 3, \dots\}$
$\mathbb{Z}$	integers	$\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$
$\mathbb{Q}$	rational numbers	$\frac{1}{2},\ -\frac{5}{3},\ 16,$ etc.
$\mathbb{R}$	real numbers	$\pi,\ e,\ -9,\ \sqrt{2},$ etc.
$\mathbb{C}$	complex numbers	$i,\ \frac{19}{2},\ \sqrt{2}-2i,$ etc.

A superscript "$^+$" restricts a set to its positive elements; for example, $\mathbb{R}^+$ denotes the set of positive real numbers. Similarly, $\mathbb{Z}^-$ denotes the set of negative integers.

4.1.2 Comparing and Combining Sets

The expression $S \subseteq T$ indicates that set S is a *subset* of set T, which means that every element of S is also an element of T. For example, $\mathbb{N} \subseteq \mathbb{Z}$ because every nonnegative integer is an integer; $\mathbb{Q} \subseteq \mathbb{R}$ because every rational number is a real number, but $\mathbb{C} \not\subseteq \mathbb{R}$ because not every complex number is a real number.

As a memory trick, think of the "$\subseteq$" symbol as like the "$\leq$" sign with the smaller set or number on the left hand side. Notice that just as $n \leq n$ for any number n, also $S \subseteq S$ for any set S.

There is also a relation, $\subset$, on sets like the "less than" relation $<$ on numbers. $S \subset T$ means that S is a subset of T, but the two are *not* equal. So just as $n \not< n$ for every number n, also $A \not\subset A$, for every set A. "$S \subset T$" is read as "S is a *strict subset* of T."

There are several basic ways to combine sets. For example, suppose

$$\begin{aligned} X &::= \{1, 2, 3\}, \\ Y &::= \{2, 3, 4\}. \end{aligned}$$

Definition 4.1.1.

- The *union* of sets A and B, denoted $A \cup B$, includes exactly the elements appearing in A or B or both. That is,

  $$x \in A \cup B \quad \text{IFF} \quad x \in A \text{ OR } x \in B.$$

  So $X \cup Y = \{1, 2, 3, 4\}$.

- The *intersection* of A and B, denoted $A \cap B$, consists of all elements that appear in *both* A and B. That is,

$$x \in A \cap B \quad \text{IFF} \quad x \in A \text{ AND } x \in B.$$

So, $X \cap Y = \{2, 3\}$.

- The *set difference* of A and B, denoted $A - B$, consists of all elements that are in A, but not in B. That is,

$$x \in A - B \quad \text{IFF} \quad x \in A \text{ AND } x \notin B.$$

So, $X - Y = \{1\}$ and $Y - X = \{4\}$.

Often all the sets being considered are subsets of a known domain of discourse, D. Then for any subset, A, of D, we define $\overline{A}$ to be the set of all elements of D *not* in A. That is,

$$\overline{A} ::= D - A.$$

The set $\overline{A}$ is called the *complement* of A. So

$$\overline{A} = \emptyset \text{ IFF } A = D.$$

For example, if the domain we're working with is the integers, the complement of the nonnegative integers is the set of negative integers:

$$\overline{\mathbb{N}} = \mathbb{Z}^-.$$

We can use complement to rephrase subset in terms of equality

$$A \subseteq B \text{ is equivalent to } A \cap \overline{B} = \emptyset.$$

4.1.3 Power Set

The set of all the subsets of a set, A, is called the *power set*, $\text{pow}(A)$, of A. So

$$B \in \text{pow}(A) \quad \text{IFF} \quad B \subseteq A.$$

For example, the elements of $\text{pow}(\{1, 2\})$ are $\emptyset, \{1\}, \{2\}$ and $\{1, 2\}$.

More generally, if A has n elements, then there are 2^n sets in $\text{pow}(A)$—see Theorem 4.5.5. For this reason, some authors use the notation 2^A instead of $\text{pow}(A)$.

4.1.4 Set Builder Notation

An important use of predicates is in *set builder notation*. We'll often want to talk about sets that cannot be described very well by listing the elements explicitly or by taking unions, intersections, etc., of easily described sets. Set builder notation often comes to the rescue. The idea is to define a *set* using a *predicate*; in particular, the set consists of all values that make the predicate true. Here are some examples of set builder notation:

$$\begin{aligned}
A &::= \{n \in \mathbb{N} \mid n \text{ is a prime and } n = 4k + 1 \text{ for some integer } k\}\\
B &::= \{x \in \mathbb{R} \mid x^3 - 3x + 1 > 0\}\\
C &::= \{a + bi \in \mathbb{C} \mid a^2 + 2b^2 \le 1\}
\end{aligned}$$

The set A consists of all nonnegative integers n for which the predicate

"n is a prime and $n = 4k + 1$ for some integer k"

is true. Thus, the smallest elements of A are:

$$5, 13, 17, 29, 37, 41, 53, 61, 73, \ldots.$$

Trying to indicate the set A by listing these first few elements wouldn't work very well; even after ten terms, the pattern is not obvious! Similarly, the set B consists of all real numbers x for which the predicate

$$x^3 - 3x + 1 > 0$$

is true. In this case, an explicit description of the set B in terms of intervals would require solving a cubic equation. Finally, set C consists of all complex numbers $a + bi$ such that:

$$a^2 + 2b^2 \le 1$$

This is an oval-shaped region around the origin in the complex plane.

4.1.5 Proving Set Equalities

Two sets are defined to be equal if they have exactly the same elements. That is, $X = Y$ means that $z \in X$ if and only if $z \in Y$, for all elements, z.[2] So, set equalities can be formulated and proved as "iff" theorems. For example:

[2]This is actually the first of the ZFC axioms for set theory mentioned at the end of Section 1.3 and discussed further in Section 7.3.2.

Theorem 4.1.2. *[*Distributive Law *for Sets] Let A, B, and C be sets. Then:*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{4.1}$$

Proof. The equality (4.1) is equivalent to the assertion that

$$z \in A \cap (B \cup C) \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) \tag{4.2}$$

for all z. Now we'll prove (4.2) by a chain of iff's.

Now we have

$$\begin{aligned}
& z \in A \cap (B \cup C) \\
& \quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \cup C) && \text{(def of } \cap) \\
& \quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \text{ OR } z \in C) && \text{(def of } \cup) \\
& \quad \text{iff} \quad (z \in A \text{ AND } z \in B) \text{ OR } (z \in A \text{ AND } z \in C) && \text{(AND distributivity Thm 3.4.1)} \\
& \quad \text{iff} \quad (z \in A \cap B) \text{ OR } (z \in A \cap C) && \text{(def of } \cap) \\
& \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) && \text{(def of } \cup)
\end{aligned}$$

■

Although the basic set operations and propositional connectives are similar, it's important not to confuse one with the other. For example, $\cup$ resembles OR, and in fact was defined directly in terms of OR:

$$x \in A \cup B \text{ is equivalent to } (x \in A \ \text{ OR } \ x \in B).$$

Similarly, $\cap$ resembles AND, and complement resembles NOT.

But if A and B are sets, writing A AND B is a type-error, since AND is an operation on truth-values, not sets. Similarly, if P and Q are propositional variables, writing $P \cup Q$ is another type-error.

The proof of Theorem 4.1.2 illustrates a general method for proving a set equality involving the basic set operations by checking that a corresponding propositional formula is valid. As a further example, from De Morgan's Law (3.11) for propositions

$$\text{NOT}(P \text{ AND } Q) \text{ is equivalent to } \overline{P} \text{ OR } \overline{Q}$$

we can derive (Problem 4.5) a corresponding De Morgan's Law for set equality:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}. \tag{4.3}$$

Despite this correspondence between two kinds of operations, it's important not to confuse propositional operations with set operations. For example, if X and Y

are sets, then it is wrong to write "X AND Y" instead of "$X \cap Y$." Applying AND to sets will cause your compiler—or your grader—to throw a type error, because an operation that is only supposed to be applied to truth values has been applied to sets. Likewise, if P and Q are propositions, then it is a type error to write "$P \cup Q$" instead of "P OR Q."

4.2 Sequences

Sets provide one way to group a collection of objects. Another way is in a *sequence*, which is a list of objects called *terms* or *components*. Short sequences are commonly described by listing the elements between parentheses; for example, (a, b, c) is a sequence with three terms.

While both sets and sequences perform a gathering role, there are several differences.

- The elements of a set are required to be distinct, but terms in a sequence can be the same. Thus, (a, b, a) is a valid sequence of length three, but $\{a, b, a\}$ is a set with two elements, not three.
- The terms in a sequence have a specified order, but the elements of a set do not. For example, (a, b, c) and (a, c, b) are different sequences, but $\{a, b, c\}$ and $\{a, c, b\}$ are the same set.
- Texts differ on notation for the *empty sequence*; we use λ for the empty sequence.

The product operation is one link between sets and sequences. A *Cartesian product* of sets, $S_1 \times S_2 \times \cdots \times S_n$, is a new set consisting of all sequences where the first component is drawn from S_1, the second from S_2, and so forth. Length two sequences are called *pairs*.[3] For example, $\mathbb{N} \times \{a, b\}$ is the set of all pairs whose first element is a nonnegative integer and whose second element is an a or a b:

$$\mathbb{N} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b), (2, a), (2, b), \ldots\}$$

A product of n copies of a set S is denoted S^n. For example, $\{0, 1\}^3$ is the set of all 3-bit sequences:

$$\{0, 1\}^3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$$

[3] Some texts call them *ordered pairs*.

4.3 Functions

4.3.1 Domains and Images

A *function* assigns an element of one set, called the *domain*, to an element of another set, called the *codomain*. The notation

$$f : A \to B$$

indicates that f is a function with domain, A, and codomain, B. The familiar notation "$f(a) = b$" indicates that f assigns the element $b \in B$ to a. Here b would be called the *value* of f at *argument* a.

Functions are often defined by formulas, as in:

$$f_1(x) ::= \frac{1}{x^2}$$

where x is a real-valued variable, or

$$f_2(y, z) ::= y\texttt{10}yz$$

where y and z range over binary strings, or

$$f_3(x, n) ::= \text{ the length } n \text{ sequence } \underbrace{(x, \dots, x)}_{n\ x\text{'s}}$$

where n ranges over the nonnegative integers.

A function with a finite domain could be specified by a table that shows the value of the function at each element of the domain. For example, a function $f_4(P, Q)$ where P and Q are propositional variables is specified by:

P	Q	$f_4(P, Q)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Notice that f_4 could also have been described by a formula:

$$f_4(P, Q) ::= [P \text{ IMPLIES } Q].$$

A function might also be defined by a procedure for computing its value at any element of its domain, or by some other kind of specification. For example, define

$f_5(y)$ to be the length of a left to right search of the bits in the binary string y until a `1` appears, so

$$\begin{aligned} f_5(0010) &= 3, \\ f_5(100) &= 1, \\ f_5(0000) &\text{ is undefined.} \end{aligned}$$

Notice that f_5 does not assign a value to any string of just `0`'s. This illustrates an important fact about functions: they need not assign a value to every element in the domain. In fact this came up in our first example $f_1(x) = 1/x^2$, which does not assign a value to 0. So in general, functions may be *partial functions*, meaning that there may be domain elements for which the function is not defined. If a function is defined on every element of its domain, it is called a *total function*.

It's often useful to find the set of values a function takes when applied to the elements in *a set* of arguments. So if $f : A \to B$, and S is a subset of A, we define $f(S)$ to be the set of all the values that f takes when it is applied to elements of S. That is,

$$f(S) ::= \{b \in B \mid f(s) = b \text{ for some } s \in S\}.$$

For example, if we let $[r, s]$ denote set of numbers in the interval from r to s on the real line, then $f_1([1, 2]) = [1/4, 1]$.

For another example, let's take the "search for a `1`" function, f_5. If we let X be the set of binary words which start with an even number of `0`'s followed by a `1`, then $f_5(X)$ would be the odd nonnegative integers.

Applying f to a set, S, of arguments is referred to as "applying f *pointwise* to S", and the set $f(S)$ is referred to as the *image* of S under f.[4] The set of values that arise from applying f to all possible arguments is called the *range* of f. That is,

$$\text{range}(f) ::= f(\text{domain}(f)).$$

Some authors refer to the codomain as the range of a function, but they shouldn't. The distinction between the range and codomain will be important later in Sections 4.5 when we relate sizes of sets to properties of functions between them.

4.3.2 Function Composition

Doing things step by step is a universal idea. Taking a walk is a literal example, but so is cooking from a recipe, executing a computer program, evaluating a formula, and recovering from substance abuse.

[4]There is a picky distinction between the function f which applies to elements of A and the function which applies f pointwise to subsets of A, because the domain of f is A, while the domain of pointwise-f is $\text{pow}(A)$. It is usually clear from context whether f or pointwise-f is meant, so there is no harm in overloading the symbol f in this way.

Abstractly, taking a step amounts to applying a function, and going step by step corresponds to applying functions one after the other. This is captured by the operation of *composing* functions. Composing the functions f and g means that first f is applied to some argument, x, to produce $f(x)$, and then g is applied to that result to produce $g(f(x))$.

Definition 4.3.1. For functions $f : A \to B$ and $g : B \to C$, the *composition*, $g \circ f$, of g with f is defined to be the function from A to C defined by the rule:

$$(g \circ f)(x) ::= g(f(x)),$$

for all $x \in A$.

Function composition is familiar as a basic concept from elementary calculus, and it plays an equally basic role in discrete mathematics.

4.4 Binary Relations

Binary relations define relations between two objects. For example, "less-than" on the real numbers relates every real number, a, to a real number, b, precisely when $a < b$. Similarly, the subset relation relates a set, A, to another set, B, precisely when $A \subseteq B$. A function $f : A \to B$ is a special case of binary relation in which an element $a \in A$ is related to an element $b \in B$ precisely when $b = f(a)$.

In this section we'll define some basic vocabulary and properties of binary relations.

Definition 4.4.1. A *binary relation*, R, consists of a set, A, called the *domain* of R, a set, B, called the *codomain* of R, and a subset of $A \times B$ called the *graph of* R.

A relation whose domain is A and codomain is B is said to be "between A and B", or "from A to B." As with functions, we write $R : A \to B$ to indicate that R is a relation from A to B. When the domain and codomain are the same set, A, we simply say the relation is "on A." It's common to use "$a \ R \ b$" to mean that the pair (a, b) is in the graph of R.[5]

Notice that Definition 4.4.1 is exactly the same as the definition in Section 4.3 of a *function*, except that it doesn't require the functional condition that, for each

[5]Writing the relation or operator symbol between its arguments is called *infix notation*. Infix expressions like "$m < n$" or "$m + n$" are the usual notation used for things like the less-then relation or the addition operation rather than prefix notation like "$< (m, n)$" or "$+(m, n)$."

domain element, a, there is *at most* one pair in the graph whose first coordinate is a. As we said, a function is a special case of a binary relation.

The “in-charge of” relation, *Chrg*, for MIT in Spring ’10 subjects and instructors is a handy example of a binary relation. Its domain, Fac, is the names of all the MIT faculty and instructional staff, and its codomain is the set, SubNums, of subject numbers in the Fall ’09–Spring ’10 MIT subject listing. The graph of *Chrg* contains precisely the pairs of the form

$$(\langle\text{instructor-name}\rangle, \langle\text{subject-num}\rangle)$$

such that the faculty member named ⟨instructor-name⟩ is in charge of the subject with number ⟨subject-num⟩ that was offered in Spring ’10. So graph(*Chrg*) contains pairs like

$$\begin{array}{ll}
(\texttt{T. Eng,} & \texttt{6.UAT}) \\
(\texttt{G. Freeman,} & \texttt{6.011}) \\
(\texttt{G. Freeman,} & \texttt{6.UAT}) \\
(\texttt{G. Freeman,} & \texttt{6.881}) \\
(\texttt{G. Freeman,} & \texttt{6.882}) \\
(\texttt{J. Guttag,} & \texttt{6.00}) \\
(\texttt{A. R. Meyer,} & \texttt{6.042}) \\
(\texttt{A. R. Meyer,} & \texttt{18.062}) \\
(\texttt{A. R. Meyer,} & \texttt{6.844}) \\
(\texttt{T. Leighton,} & \texttt{6.042}) \\
(\texttt{T. Leighton,} & \texttt{18.062}) \\
\quad\vdots &
\end{array} \tag{4.4}$$

Some subjects in the codomain, SubNums, do not appear among this list of pairs—that is, they are not in range(*Chrg*). These are the Fall term-only subjects. Similarly, there are instructors in the domain, Fac, who do not appear in the list because they are not in charge of any Spring term subjects.

4.4.1 Relation Diagrams

Some standard properties of a relation can be visualized in terms of a diagram. The diagram for a binary relation, R, has points corresponding to the elements of the domain appearing in one column (a very long column if domain(R) is infinite). All the elements of the codomain appear in another column which we’ll usually picture as being to the right of the domain column. There is an arrow going from a point, a, in the lefthand, domain column to a point, b, in the righthand, codomain column, precisely when the corresponding elements are related by R. For example, here are diagrams for two functions:

$$\begin{tikzcd}[row sep=small]
A & & B \\
a \arrow[rr] & & 1 \\
b \arrow[rrd] & & 2 \\
c \arrow[rrd] & & 3 \\
d \arrow[rruu] & & 4 \\
e \arrow[rruu] & &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}[row sep=small]
A & & B \\
a \arrow[rr] & & 1 \\
b \arrow[rrd] & & 2 \\
c \arrow[rrdd] & & 3 \\
d \arrow[rruu] & & 4 \\
 & & 5
\end{tikzcd}$$

Being a function is certainly an important property of a binary relation. What it means is that every point in the domain column has *at most one arrow coming out of it*. So we can describe being a function as the "$\leq$ 1 arrow out" property. There are four more standard properties of relations that come up all the time. Here are all five properties defined in terms of arrows:

Definition 4.4.2. A binary relation, R, is:

- a *function* when it has the [$\leq$ 1 arrow **out**] property.
- *surjective* when it has the [$\geq$ 1 arrows **in**] property. That is, every point in the righthand, codomain column has at least one arrow pointing to it.
- *total* when it has the [$\geq$ 1 arrows **out**] property.
- *injective* when it has the [$\leq$ 1 arrow **in**] property.
- *bijective* when it has both the [= 1 arrow **out**] and the [= 1 arrow **in**] property.

From here on, we'll stop mentioning the arrows in these properties and for example, just write [$\leq$ 1 in] instead of [$\leq$ 1 arrows in].

So in the diagrams above, the relation on the left has the [= 1 out] and [$\geq$ 1 in] properties, which means it is a total, surjective function. But it does not have the [$\leq$ 1 in] property because element 3 has two arrows going into it; it is not injective.

The relation on the right has the [= 1 out] and [$\leq$ 1 in] properties, which means it is a total, injective function. But it does not have the [$\geq$ 1 in] property because element 4 has no arrow going into it; it is not surjective.

The arrows in a diagram for R correspond, of course, exactly to the pairs in the graph of R. Notice that the arrows alone are not enough to determine, for example, if R has the [$\geq$ 1 out], total, property. If all we knew were the arrows, we wouldn't know about any points in the domain column that had no arrows out. In other words, graph(R) alone does not determine whether R is total: we also need to know what domain(R) is.

Example 4.4.3. The function defined by the formula $1/x^2$ has the [≥ 1 out] property if its domain is $\mathbb{R}^+$, but not if its domain is some set of real numbers including 0. It has the [$= 1$ in] and [$= 1$ out] property if its domain and codomain are both $\mathbb{R}^+$, but it has neither the [≤ 1 in] nor the [≥ 1 out] property if its domain and codomain are both $\mathbb{R}$.

4.4.2 Relational Images

The idea of the image of a set under a function extends directly to relations.

Definition 4.4.4. The *image* of a set, Y, under a relation, R, written $R(Y)$, is the set of elements of the codomain, B, of R that are related to some element in Y. In terms of the relation diagram, $R(Y)$ is the set of points with an arrow coming in that starts from some point in Y.

For example, the set of subject numbers that Meyer is in charge of in Spring ’10 is exactly *Chrg*(A. Meyer). To figure out what this is, we look for all the arrows in the *Chrg* diagram that start at “A. Meyer,” and see which subject-numbers are at the other end of these arrows. Looking at the list (4.4) of pairs in graph(*Chrg*), we see that these subject-numbers are {6.042, 18.062, 6.844}. Similarly, to find the subject numbers that either Freeman or Eng are in charge of, we can collect all the arrows that start at either “G. Freeman,” or “T. Eng” and, again, see which subject-numbers are at the other end of these arrows. This is *Chrg*({G. Freeman, T. Eng}). Looking again at the list (4.4), we see that

$$\mathit{Chrg}(\{\text{G. Freeman}, \text{T. Eng}\}) = \{6.011, 6.881, 6.882, 6.\text{UAT}\}$$

Finally, Fac is the set of all in-charge instructors, so *Chrg*(Fac) is the set of all the subjects listed for Spring ’10.

Inverse Relations and Images

Definition 4.4.5. The *inverse*, R^{-1} of a relation $R : A \to B$ is the relation from B to A defined by the rule

$$b \; R^{-1} \; a \quad \text{IFF} \quad a \; R \; b.$$

In other words, R^{-1} is the relation you get by reversing the direction of the arrows in the diagram of R.

Definition 4.4.6. The image of a set under the relation, R^{-1}, is called the *inverse image* of the set. That is, the inverse image of a set, X, under the relation, R, is defined to be $R^{-1}(X)$.

Continuing with the in-charge example above, the set of instructors in charge of 6.UAT in Spring '10 is exactly the inverse image of {6.UAT} under the *Chrg* relation. From the list (4.4), we see that Eng and Freeman are both in charge of 6.UAT, that is,

$$\{\text{T. Eng}, \text{D. Freeman}\} \subseteq \mathit{Chrg}^{-1}(\{\text{6.UAT}\}).$$

We can't assert equality here because there may be additional pairs further down the list showing that additional instructors are co-incharge of 6.UAT.

Now let Intro be the set of introductory course 6 subject numbers. These are the subject numbers that start with "6.0." So the set of names of the instructors who were in-charge of introductory course 6 subjects in Spring '10, is $\mathit{Chrg}^{-1}(\text{Intro})$. From the part of the *Chrg* list shown in (4.4), we see that Meyer, Leighton, Freeman, and Guttag were among the instructors in charge of introductory subjects in Spring '10. That is,

$$\{\text{Meyer}, \text{Leighton}, \text{Freeman}, \text{Guttag}\} \subseteq \mathit{Chrg}^{-1}(\text{Intro}).$$

Finally, $\mathit{Chrg}^{-1}(\text{SubNums})$, is the set of all instructors who were in charge of a subject listed for Spring '10.

4.5 Finite Cardinality

A finite set is one that has only a finite number of elements. This number of elements is the "size" or *cardinality* of the set:

Definition 4.5.1. If A is a finite set, the *cardinality* of A, written $|A|$, is the number of elements in A.

A finite set may have no elements (the empty set), or one element, or two elements,..., so the cardinality of finite sets is always a nonnegative integer.

Now suppose $R : A \to B$ is a function. This means that every element of A contributes at most one arrow to the diagram for R, so the number of arrows is at most the number of elements in A. That is, if R is a function, then

$$|A| \geq \#\text{arrows}.$$

If R is also surjective, then every element of B has an arrow into it, so there must be at least as many arrows in the diagram as the size of B. That is,

$$\#\text{arrows} \geq |B|.$$

Combining these inequalities implies that if R is a surjective function, then $|A| \geq |B|$.

In short, if we write A surj B to mean that there is a surjective function from A to B, then we've just proved a lemma: if A surj B for finite sets A, B, then $|A| \geq |B|$. The following definition and lemma lists this statement and three similar rules relating domain and codomain size to relational properties.

Definition 4.5.2. Let A, B be (not necessarily finite) sets. Then

1. A surj B iff there is a surjective *function* from A to B.
2. A inj B iff there is an injective *total* relation from A to B.
3. A bij B iff there is a bijection from A to B.

Lemma 4.5.3. *For finite sets A, B:*

1. *If A* surj *B, then $|A| \geq |B|$.*
2. *If A* inj *B, then $|A| \leq |B|$.*
3. *If A* bij *B, then $|A| = |B|$.*

Proof. We've already given an "arrow" proof of implication 1. Implication 2. follows immediately from the fact that if R has the [$\leq$ 1 out], function property, and the [$\geq$ 1 in], surjective property, then R^{-1} is total and injective, so A surj B iff B inj A. Finally, since a bijection is both a surjective function and a total injective relation, implication 3. is an immediate consequence of the first two. ■

Lemma 4.5.3.1. has a converse: if the size of a finite set, A, is greater than or equal to the size of another finite set, B, then it's always possible to define a surjective function from A to B. In fact, the surjection can be a total function. To see how this works, suppose for example that

$$A = \{a_0, a_1, a_2, a_3, a_4, a_5\}$$
$$B = \{b_0, b_1, b_2, b_3\}.$$

Then define a total function $f : A \to B$ by the rules

$$f(a_0) ::= b_0,\ f(a_1) ::= b_1,\ f(a_2) ::= b_2,\ f(a_3) = f(a_4) = f(a_5) ::= b_3.$$

More concisely,

$$f(a_i) ::= b_{\min(i,3)},$$

for $0 \le i \le 5$. Since $5 \ge 3$, this f is a surjection.

So we have figured out that if A and B are finite sets, then $|A| \ge |B|$ *if and only if* A surj B. All told, this argument wraps up the proof of a theorem that summarizes the whole finite cardinality story:

Theorem 4.5.4. *[Mapping Rules] For* finite *sets, A, B,*

$$|A| \ge |B| \quad \textit{iff} \quad A \text{ surj } B, \tag{4.5}$$
$$|A| \le |B| \quad \textit{iff} \quad A \text{ inj } B, \tag{4.6}$$
$$|A| = |B| \quad \textit{iff} \quad A \text{ bij } B, \tag{4.7}$$

4.5.1 How Many Subsets of a Finite Set?

As an application of the bijection mapping rule (4.7), we can give an easy proof of:

Theorem 4.5.5. *There are 2^n subsets of an n-element set. That is,*

$$|A| = n \quad \textit{implies} \quad |\operatorname{pow}(A)| = 2^n.$$

For example, the three-element set $\{a_1, a_2, a_3\}$ has eight different subsets:

$$\begin{array}{cccc} \emptyset & \{a_1\} & \{a_2\} & \{a_1, a_2\} \\ \{a_3\} & \{a_1, a_3\} & \{a_2, a_3\} & \{a_1, a_2, a_3\} \end{array}$$

Theorem 4.5.5 follows from the fact that there is a simple bijection from subsets of A to $\{0,1\}^n$, the n-bit sequences. Namely, let $a_1, a_2, \ldots, a_n$ be the elements of A. The bijection maps each subset of $S \subseteq A$ to the bit sequence $(b_1, \ldots, b_n)$ defined by the rule that

$$b_i = 1 \quad \text{iff} \quad a_i \in S.$$

For example, if $n = 10$, then the subset $\{a_2, a_3, a_5, a_7, a_{10}\}$ maps to a 10-bit sequence as follows:

$$\begin{array}{rcccccccccccc} \text{subset:} & \{ & & a_2, & a_3, & & a_5, & & a_7, & & & a_{10} & \} \\ \text{sequence:} & (& 0, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1 &) \end{array}$$

Now by bijection case of the Mapping Rules 4.5.4.(4.7),

$$|\operatorname{pow}(A)| = |\{0,1\}^n|.$$

But every computer scientist knows[6] that there are 2^n n-bit sequences! So we've proved Theorem 4.5.5!

[6]In case you're someone who doesn't know how many n-bit sequences there are, you'll find the 2^n explained in Section 14.2.2.

Problems for Section 4.1

Practice Problems

Problem 4.1.
For any set A, let $\text{pow}(A)$ be its *power set*, the set of all its subsets; note that A is itself a member of $\text{pow}(A)$. Let $\emptyset$ denote the empty set.

(a) The elements of $\text{pow}(\{1, 2\})$ are:

(b) The elements of $\text{pow}(\{\emptyset, \{\emptyset\}\})$ are:

(c) How many elements are there in $\text{pow}(\{1, 2, \dots, 8\})$?

Problem 4.2.
Express each of the following assertions about sets by a formula of set theory.[7]

(a) $x = \emptyset$.

(b) $x = \{y, z\}$.

(c) $x \subseteq y$. (x is a subset of y that might equal y.)

Now we can explain how to express "x is a proper subset of y" as a set theory formula using things we already know how to express. Namely, letting "$x \neq y$" abbreviate $\text{NOT}(x = y)$, the expression

$$(x \subseteq y \ \text{ AND } \ x \neq y),$$

describes a formula of set theory that means $x \subset y$.

From here on, feel free to use any previously expressed property in describing formulas for the following:

(d) $x = y \cup z$.

(e) $x = y - z$.

(f) $x = \text{pow}(y)$.

(g) $x = \bigcup_{z \in y} z$.

This means that y is supposed to be a collection of sets, and x is the union of all of them. A more concise notation for "$\bigcup_{z \in y} z$' is simply "$\bigcup y$."

[7]See Section 7.3.2.

Class Problems

Problem 4.3.
Set Formulas and Propositional Formulas.
(a) Verify that the propositional formula $(P \text{ AND } \overline{Q}) \text{ OR } (P \text{ AND } Q)$ is equivalent to P.

(b) Prove that
$$A = (A - B) \cup (A \cap B)$$
for all sets, A, B, by showing
$$x \in A \text{ IFF } x \in (A - B) \cup (A \cap B)$$
for all elements, x, using the equivalence of part (a) in a chain of IFF's.

Problem 4.4.
Prove

Theorem (Distributivity of union over intersection)**.**
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \tag{4.8}$$
for all sets, A, B, C, by using a chain of iff's to show that
$$x \in A \cup (B \cap C) \text{ IFF } x \in (A \cup B) \cap (A \cup C)$$
for all elements, x. You may assume the corresponding propositional equivalence Theorem 3.4.2.

Problem 4.5.
Prove De Morgan's Law for set equality
$$\overline{A \cap B} = \overline{A} \cup \overline{B}. \tag{4.9}$$
by showing with a chain of IFF's that $x \in$ the left hand side of (4.9) iff $x \in$ the right hand side. You may assume the propositional version (3.11) of De Morgan's Law.

Problem 4.6.
Powerset Properties.
Let A and B be sets.

(a) Prove that

$$\text{pow}(A \cap B) = \text{pow}(A) \cap \text{pow}(B).$$

(b) Prove that

$$\text{pow}(A) \cup \text{pow}(B) \subseteq \text{pow}(A \cup B),$$

with equality holding iff one of A or B is a subset of the other.

Problem 4.7.
Subset take-away[8] is a two player game played with a finite set, A, of numbers. Players alternately choose nonempty subsets of A with the conditions that a player may not choose

- the whole set A, or
- any set containing a set that was named earlier.

The first player who is unable to move loses the game.

For example, if the size of A is one, then there are no legal moves and the second player wins. If A has exactly two elements, then the only legal moves are the two one-element subsets of A. Each is a good reply to the other, and so once again the second player wins.

The first interesting case is when A has three elements. This time, if the first player picks a subset with one element, the second player picks the subset with the other two elements. If the first player picks a subset with two elements, the second player picks the subset whose sole member is the third element. In both cases, these moves lead to a situation that is the same as the start of a game on a set with two elements, and thus leads to a win for the second player.

Verify that when A has four elements, the second player still has a winning strategy.[9]

Homework Problems

Problem 4.8.
Let A, B, and C be sets. Prove that:

$$A \cup B \cup C = (A - B) \cup (B - C) \cup (C - A) \cup (A \cap B \cap C). \tag{4.10}$$

[8]From Christenson & Tilford, *David Gale's Subset Takeaway Game, American Mathematical Monthly, Oct. 1997*

[9]David Gale worked out some of the properties of this game and conjectured that the second player wins the game for any set A. This remains an open problem.

Hint: P OR Q OR R is equivalent to

$$(P \text{ AND } \overline{Q}) \text{ OR } (Q \text{ AND } \overline{R}) \text{ OR } (R \text{ AND } \overline{P}) \text{ OR } (P \text{ AND } Q \text{ AND } R).$$

Problem 4.9.
Union distributes over the intersection of two sets:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \tag{4.11}$$

(see Problem 4.4).

Use (4.11) and the Well Ordering Principle to prove the Distributive Law of union over the intersection of n sets:

$$A \cup (B_1 \cap \cdots \cap B_{n-1} \cap B_n) = (A \cup B_1) \cap \cdots \cap (A \cup B_{n-1}) \cap (A \cup B_n) \tag{4.12}$$

Extending formulas to an arbitrary number of terms is a common (if mundane) application of the WOP.

Exam Problems

Problem 4.10.
You've seen how certain set identities follow from corresponding propositional equivalences. For example, you proved by a chain of iff's that

$$(A - B) \cup (A \cap B) = A$$

using the fact that the propositional formula $(P \text{ AND } \overline{Q}) \text{ OR } (P \text{ AND } Q)$ is equivalent to P.

State a similar propositional equivalence that would justify the key step in a proof for the following set equality organized as a chain of iff's:

$$\overline{A - B} = \left(\overline{A} - \overline{C}\right) \cup (B \cap C) \cup \left(\left(\overline{A} \cup B\right) \cap \overline{C}\right) \tag{4.13}$$

(You are *not* being asked to write out an iff-proof of the equality or to write out a proof of the propositional equivalence. Just state the equivalence.)

Problem 4.11.
You've seen how certain set identities follow from corresponding propositional equivalences. For example, you proved by a chain of iff's that

$$(A - B) \cup (A \cap B) = A$$

using the fact that the propositional formula $(P \text{ AND } \overline{Q}) \text{ OR } (P \text{ AND } Q)$ is equivalent to P.

State a similar propositional equivalence that would justify the key step in a proof for the following set equality organized as a chain of iff's:

$$\overline{A \cap B \cap C} = \overline{A} \cup (\overline{B} - \overline{A}) \cup \overline{C}.$$

(You are *not* being asked to write out an iff-proof of the equality or to write out a proof of the propositional equivalence. Just state the equivalence.)

Problems for Section 4.2

Homework Problems

Problem 4.12.
Prove that for any sets A, B, C, and D, if the Cartesian products $A \times B$ and $C \times D$ are disjoint, then either A and C are disjoint or B and D are disjoint.

Problem 4.13. **(a)** Give a simple example where the following result fails, and briefly explain why:

False Theorem. *For sets A, B, C, and D, let*

$$\begin{aligned} L &::= (A \cup B) \times (C \cup D), \\ R &::= (A \times C) \cup (B \times D). \end{aligned}$$

Then $L = R$.

(b) Identify the mistake in the following proof of the False Theorem.

Bogus proof. Since L and R are both sets of pairs, it's sufficient to prove that $(x, y) \in L \longleftrightarrow (x, y) \in R$ for all x, y.

The proof will be a chain of iff implications:

$$\begin{array}{ll} & (x, y) \in R \\ \text{iff} & (x, y) \in (A \times C) \cup (B \times D) \\ \text{iff} & (x, y) \in A \times C, \text{ or } (x, y) \in B \times D \\ \text{iff} & (x \in A \text{ and } y \in C) \text{ or else } (x \in B \text{ and } y \in D) \\ \text{iff} & \text{either } x \in A \text{ or } x \in B, \text{ and either } y \in C \text{ or } y \in D \\ \text{iff} & x \in A \cup B \text{ and } y \in C \cup D \\ \text{iff} & (x, y) \in L. \end{array}$$

■

(c) Fix the proof to show that $R \subseteq L$.

Problem 4.14.
A *binary word* is a finite sequence of 0's and 1's. For example, $(1, 1, 0)$ and (1) are words of length three and one, respectively. We usually omit the parentheses and commas in the descriptions of words, so the preceding binary words would just be written as 110 and 1.

The basic operation of placing one word immediately after another is called *concatentation*. For example, the concatentation of 110 and 1 is 1101, and the concatentation of 110 with itself is 110110.

We can extend this basic operation on words to an operation on *sets* of words. To emphasize the distinction between a word and a set of words, from now on we'll refer to a set of words as a *language*. Now if R and S are languages, then $R \cdot S$ is the language consisting of all the words you can get by concatenating a word from R with a word from S. That is,

$$R \cdot S ::= \{rs \mid r \in R \text{ AND } s \in S\}.$$

For example,

$$\{0, 00\} \cdot \{00, 000\} = \{000, 0000, 00000\}$$

Another example is $D \cdot D$, abbreviated as D^2, where $D ::= \{1, 0\}$ is just the two binary digits.

$$D^2 = \{00, 01, 10, 11\}.$$

In other words, D^2 is the language consisting of all the length two words. More generally, D^n will be the language of length n words.

If S is a language, the language you can get by concatenating any number of copies of words in S is called S^*—pronounced "S star." (By convention, the empty word, λ, always included in S^*.) For example, $\{0, 11\}^*$ is the language consisting of all the words you can make by stringing together 0's and 11's. This language could also be described as consisting of the words whose blocks of 1's are always of even length. Another example is $(D^2)^*$, which consists of all the even length words. Finally, the language, B, of *all* binary words is just D^*.

A language is called *concatenation-definable* (*c-d*) if it can be constructed by starting with finite languages and then applying the operations of concatenation, union, and complement (relative to B) to these languages a finite number of times.[10] Note that the *-operation is *not* allowed. For this reason, the c-d languages are also called the “star-free languages,” [32].

Lots of interesting languages turn out to be concatenation-definable, but some

[10] We can assign to each c-d language a *count* which bounds the number of the allowed operations (Union, Concatenation, and Complement) it takes to make it.

Since finite languages are given to be c-d, they are the 0-count languages. For example,

- $\{00, 111\}$,
- the words of length $\leq 10^{10}$, and
- the empty language, $\emptyset$,

are all 0-count.

We get a 1-count language by applying one of the operations to a 0-count language. So applying the complement operation to each of the above 0-count languages gives the following 1-count languages:

- $\overline{\{00, 111\}}$, the language of all binary words except `00` and `111`,
- the words of length $> 10^{10}$, and
- the language B of all words.

These languages are all infinite, so none of them are 0-count.

Notice that you don't get anything new by using the Union operation to combine two 0-count languages, since the union of finite sets is finite. Likewise, you don't get anything new by concatenating two 0-count languages because the Concatenation of two finite languages is finite—if R and S are finite languages respectively containing n and m words, then $R \cdot S$ contains at most mn words. (Exercise, give an example where $R \cdot S$ contains *fewer* than mn words.)

So the 1-count languages that are not 0-count are precisely those that come from complementing a finite language. That is, they are the languages that include all but a finite number of words.

We can apply Concatenation to a 0-count and a 1-count language to get a 2-count language. For example,

$$\{\texttt{00}, \texttt{111}\} \cdot B$$

is a 2-count language consisting of all the words that start with either 00 or 111. Notice that this language is not 0-count or 1-count, since both it and its complement are infinite.

Doing a concatenation of the 1-count language B with this 2-count language, gives a $1+1+2 = 4$-count language

$$B \cdot \{00, 111\} \cdot B$$

which consists of all the words that have either two consecutive `0`'s or three consecutive `1`'s. We don't know at this point whether this language is also 3-count, or even 2-count, because we haven't ruled out the possibility that it could be built using fewer than 4 operations (though we don't think it can).

Now doing a complement of this 4-count language give a 5-count language consisting of all the words in which

- every occurrence of `0` is followed by a `1`, except for a possible `0` at the end of the word, and also
- every occurrence of`11` is followed by a `0`, except for a possible `11` at the end of the word.

The c-d languages are precisely the languages that are n-count for some nonnegative integer n.

very simple languages are not. This problem ends with the conclusion that the language $\{\texttt{00}\}^*$ of even length words whose bits are all $\texttt{0}$'s is not a c-d language.

(a) Show that if R and S are c-d, then so is $R \cap S$.

Now we can show that the set B of all binary words is c-d as follows. Let u and v be any two different binary words. Then $\{u\} \cap \{v\}$ equals the empty set. But $\{u\}$ and $\{v\}$ are c-d by definition, so by part (a), the empty set is c-d and therefore so is $\overline{\emptyset} = B$.

Now a more interesting example of a c-d set is language of all binary words that include three consecutive $\texttt{1}$'s:

$$B\texttt{111}B.$$

Notice that the proper expression here is "$B \cdot \{\texttt{111}\} \cdot B$." But it causes no confusion and helps readability to omit the dots in concatenations and the curly braces for sets with one element.

(b) Show that the language consisting of the binary words that start with $\texttt{0}$ and end with $\texttt{1}$ is c-d.

(c) Show that $\texttt{0}^*$ is c-d.

(d) Show that $\{\texttt{01}\}^*$ is c-d.

Let's say a language S is $\texttt{0}$-*finite* when it includes only a finite number of words whose bits are all $\texttt{0}$'s, that is, when $S \cap \texttt{0}^*$ is a finite set of words. A langauge S is $\texttt{0}$-*boring*—boring, for short—when either S or $\overline{S}$ is $\texttt{0}$-finite.

(e) Explain why $\{\texttt{00}\}^*$ is not boring.

(f) Verify that if R and S are boring, then so is $R \cup S$.

(g) Verify that if R and S are boring, then so is $R \cdot S$.

Hint: By cases: whether R and S are both $\texttt{0}$-finite, whether R or S contains no all-$\texttt{0}$ words at all (including the empty word λ), and whether neither of these cases hold.

(h) Explain why all c-d languages are boring.

So we have proved that the set $(\texttt{00})^*$ of even length all-$\texttt{0}$ words is not a c-d language.

Problems for Section 4.4

Practice Problems

Problem 4.15.
The *inverse*, R^{-1}, of a binary relation, R, from A to B, is the relation from B to A defined by:

$$b \; R^{-1} \; a \quad \text{iff} \quad a \; R \; b.$$

In other words, you get the diagram for R^{-1} from R by "reversing the arrows" in the diagram describing R. Now many of the relational properties of R correspond to different properties of R^{-1}. For example, R is *total* iff R^{-1} is a *surjection*.

Fill in the remaining entries is this table:

R is	iff R^{-1} is
total	a surjection
a function	
a surjection	
an injection	
a bijection	

Hint: Explain what's going on in terms of "arrows" from A to B in the diagram for R.

Problem 4.16.
Describe a total injective function [$= 1$ out], [≤ 1 in,] from $\mathbb{R} \to \mathbb{R}$ that is not a bijection.

Problem 4.17.
For a binary relation, $R : A \to B$, some properties of R can be determined from just the arrows of R, that is, from graph(R), and others require knowing if there are elements in the domain, A, or the codomain, B, that don't show up in graph(R). For each of the following possible properties of R, indicate whether it is always determined by

1. graph(R) alone,
2. graph(R) and A alone,
3. graph(R) and B alone,

4. all three parts of R.

Properties:

(a) surjective

(b) injective

(c) total

(d) function

(e) bijection

Problem 4.18.
For each of the following real-valued functions on the real numbers, indicate whether it is a bijection, a surjection but not a bijection, an injection but not a bijection, or neither an injection nor a surjection.

(a) $x \to x + 2$

(b) $x \to 2x$

(c) $x \to x^2$

(d) $x \to x^3$

(e) $x \to \sin x$

(f) $x \to x \sin x$

(g) $x \to e^x$

Problem 4.19.
Let $f : A \to B$ and $g : B \to C$ be functions and $h : A \to C$ be their composition, namely, $h(a) ::= g(f(a))$ for all $a \in A$.

(a) Prove that if f and g are surjections, then so is h.

(b) Prove that if f and g are bijections, then so is h.

(c) If f is a bijection, then so is f^{-1}.

Problem 4.20.
Give an example of a relation R that is a total injective function from a set A to itself but is not a bijection.

Problem 4.21.
Let $R : A \to B$ be a binary relation. Each of the following formulas expresses the fact that R has a familiar relational "arrow" property such as being surjective or being a function.

Identify the relational property expressed by each of the following relational expressions. Explain your reasoning.

(a) $R \circ R^{-1} \subseteq \text{Id}_B$

(b) $R^{-1} \circ R \subseteq \text{Id}_A$

(c) $R^{-1} \circ R \supseteq \text{Id}_A$

(d) $R \circ R^{-1} \supseteq \text{Id}_B$

Class Problems

Problem 4.22. (a) Prove that if A surj B and B surj C, then A surj C.

(b) Explain why A surj B iff B inj A.

(c) Conclude from (a) and (b) that if A inj B and B inj C, then A inj C.

(d) Explain why A inj B iff there is a total injective *function* ($[= 1$ out, ≤ 1 in$]$) from A to B. [11]

Problem 4.23.
Five basic properties of binary relations $R : A \to B$ are:

1. R is a surjection $[\geq 1$ in$]$
2. R is an injection $[\leq 1$ in$]$
3. R is a function $[\geq 1$ out$]$
4. R is total $[\geq 1$ out$]$

[11] The official definition of inj is with a total injective *relation* ($[\geq 1$ out, ≤ 1 in$]$)

5. R is empty [= 0 out]

Below are some assertions about a relation R. For each assertion, write the numbers of all the properties above that the relation R must have; write "none" if R might not have any of these properties. For example, you should write "(1), (4)" next to the first assertion.

Variables $a, a_1, \ldots$ range over A and $b, b_1, \ldots$ range over B.

(a) $\forall a\, \forall b.\ a\ R\ b.$ (1), (4)

(b) NOT$(\forall a\, \forall b.\ a\ R\ b)$.

(c) $\forall a\, \forall b.\ QNOT(a\ R\ b)$.

(d) $\forall a\, \exists b.\ a\ R\ b$.

(e) $\forall b\, \exists a.\ a\ R\ b$.

(f) R is a bijection.

(g) $\forall a\, \exists b_1\ a\ R\ b_1 \bigwedge \forall b.\ a\ R\ b$ IMPLIES $b = b_1$.

(h) $\forall a, b.\ a\ R\ b$ OR $a \neq b$.

(i) $\forall b_1, b_2, a.\ (a\ R\ b_1$ AND $a\ R\ b_2)$ IMPLIES $b_1 = b_2$.

(j) $\forall a_1, a_2, b.\ (a_1\ R\ b$ AND $a_2\ R\ b)$ IMPLIES $a_1 = a_2$.

(k) $\forall a_1, a_2, b_1, b_2.\ (a_1\ R\ b_1$ AND $a_2\ R\ b_2$ AND $a_1 \neq a_2)$ IMPLIES $b_1 \neq b_2$.

(l) $\forall a_1, a_2, b_1, b_2.\ (a_1\ R\ b_1$ AND $a_2\ R\ b_2$ AND $b_1 \neq b_2)$ IMPLIES $a_1 \neq a_2$.

Homework Problems

Problem 4.24.
Let $f : A \to B$ and $g : B \to C$ be functions.

(a) Prove that if the composition $g \circ f$ is a bijection, then f is a total injection and g is a surjection.

(b) Show there is a total injection, f, and a bijection, g, such that $g \circ f$ is not a bijection.

Problem 4.25.
Let A, B, and C be nonempty sets, and let $f : B \to C$ and $g : A \to B$ be

functions. Let $h ::= f \circ g$ be the composition function of f and g, namely, the function with domain A and range C such that $h(x) = f(g(x))$.

(a) Prove that if h is surjective and f is total and injective, then g must be surjective.

Hint: contradiction.

(b) Suppose that h is injective and f is total. Prove that g must be injective and provide a counterexample showing how this claim could fail if f was *not* total.

Problem 4.26.
Let A, B, and C be sets, and let $f : B \to C$ and $g : A \to B$ be functions. Let $h : A \to C$ be the composition, $f \circ g$, that is, $h(x) ::= f(g(x))$ for $x \in A$. Prove or disprove the following claims:

(a) If h is surjective, then f must be surjective.

(b) If h is surjective, then g must be surjective.

(c) If h is injective, then f must be injective.

(d) If h is injective and f is total, then g must be injective.

Problem 4.27.
Let R be a binary relation on a set D. Let x, y be variables ranging over D. Circle the expressions below whose meaning is that R is an *injection* [$\leq$ 1 in]. Remember R is a not necessarily total or a function.

1. $R(x) = R(y)$ IMPLIES $x = y$
2. $R(x) \cap R(y) = \emptyset$ IMPLIES $x \neq y$
3. $R(x) \cap R(y) \neq \emptyset$ IMPLIES $x \neq y$
4. $R(x) \cap R(y) \neq \emptyset$ IMPLIES $x = y$
5. $R^{-1}(R(x)) = \{x\}$
6. $R^{-1}(R(x)) \subseteq \{x\}$
7. $R^{-1}(R(x)) \supseteq \{x\}$
8. $R(R^{-1}(x)) = x$

Problem 4.28.
The language of sets and relations may seem remote from the practical world of programming, but in fact there is a close connection to *relational databases*, a very popular software application building block implemented by such software packages as MySQL. This problem explores the connection by considering how to manipulate and analyze a large data set using operators over sets and relations. Systems like MySQL are able to execute very similar high-level instructions efficiently on standard computer hardware, which helps programmers focus on high-level design.

Consider a basic Web search engine, which stores information on Web pages and processes queries to find pages satisfying conditions provided by users. At a high level, we can formalize the key information as:

- A set P of *pages* that the search engine knows about
- A binary relation L (for *link*) over pages, defined such that $p_1 \; L \; p_2$ iff page p_1 links to p_2
- A set E of *endorsers*, people who have recorded their opinions about which pages are high-quality
- A binary relation R (for *recommends*) between endorsers and pages, such that $e \; R \; p$ iff person e has recommended page p
- A set W of *words* that may appear on pages
- A binary relation M (for *mentions*) between pages and words, where $p \; M \; w$ iff word w appears on page p

Each part of this problem describes an intuitive, informal query over the data, and your job is to produce a single expression using the standard set and relation operators, such that the expression can be interpreted as answering the query correctly, for any data set. Your answers should use only the set and relation symbols given above, in addition to terms standing for constant elements of E or W, plus the following operators introduced in the text:

- set union, $\cup$.
- set intersection, $\cap$.
- set difference, $-$.
- relational image—for example, $R(A)$ for some set A, or $R(a)$ for some specific element a.

- relational inverse $^{-1}$.
- ...and one extra: *relational composition* which generalizes composition of functions
  $$a\ (R \circ S)\ c ::= \exists b \in B.\, (a\ S\ b) \text{ AND } (b\ R\ c).$$
  In other words, a is related to c in $R \circ S$ if starting at a you can follow an S arrow to the start of an R arrow and then follow the R arrow to get to c.[12]

Here is one worked example to get you started:

- **Search description:** The set of pages containing the word "logic"
- **Solution expression:** M^{-1}("logic")

Find similar solutions for each of the following searches:

(a) The set of pages containing the word "logic" but not the word "predicate"

(b) The set of pages containing the word "set" that have been recommended by "Meyer"

(c) The set of endorsers who have recommended pages containing the word "algebra"

(d) The relation that relates endorser e and word w iff e has recommended a page containing w

(e) The set of pages that have at least one incoming or outgoing link

(f) The relation that relates word w and page p iff w appears on a page that links to p

(g) The relation that relates word w and endorser e iff w appears on a page that links to a page that e recommends

(h) The relation that relates pages p_1 and p_2 iff p_2 can be reached from p_1 by following a sequence of exactly 3 links

[12]Note the reversal of R and S in the definition; this is to make relational composition work like function composition. For functions, $f \circ g$ means you apply g first. That is, if we let h be $f \circ g$, then $h(x) = f(g(x))$.

Exam Problems

Problem 4.29.
Let A be the set containing the five sets: $\{a\}, \{b,c\}, \{b,d\}, \{a,e\}, \{e,f\}$, and let B be the set containing the three sets: $\{a,b\}, \{b,c,d\}, \{e,f\}$. Let R be the “is subset of” binary relation from A to B defined by the rule:

$$X \mathrel{R} Y \quad \text{IFF} \quad X \subseteq Y.$$

(a) Fill in the arrows so the following figure describes the graph of the relation, R:

A	arrows	B
$\{a\}$		
		$\{a,b\}$
$\{b,c\}$		
		$\{b,c,d\}$
$\{b,d\}$		
		$\{e,f\}$
$\{a,e\}$		
$\{e,f\}$		

(b) Circle the properties below possessed by the relation R:

function total injective surjective bijective

(c) Circle the properties below possessed by the relation R^{-1}:

function total injective surjective bijective

Problem 4.30. **(a)** Five assertions about a binary relation $R : A \to B$ are bulleted below. There are nine predicate formulas that express some of these assertions. Write the numbers of the formulas next to the assertions they express. For example, you should write “4” next to the last assertion, since formula (4) expresses the assertion that R is the identity relation.

Variables $a, a_1, \ldots$ range over the domain A and $b, b_1, \ldots$ range over the codomain B. More than one formula may express one assertion.

- R is a surjection
- R is an injection
- R is a function
- R is total
- R is the identity relation.

1. $\forall b. \exists a.\ a\ R\ b.$
2. $\forall a. \exists b.\ a\ R\ b.$
3. $\forall a.\ a\ R\ a.$
4. $\forall a, b.\ a\ R\ b$ IFF $a = b.$
5. $\forall a, b.\ a\ R\ b$ OR $a \neq b.$
6. $\forall b_1, b_2, a.\ (a\ R\ b_1$ AND $a\ R\ b_2)$ IMPLIES $b_1 = b_2.$
7. $\forall a_1, a_2, b.\ (a_1\ R\ b$ AND $a_2\ R\ b)$ IMPLIES $a_1 = a_2.$
8. $\forall a_1, a_2, b_1, b_2.\ (a_1\ R\ b_1$ AND $a_2\ R\ b_2$ AND $a_1 \neq a_2)$ IMPLIES $b_1 \neq b_2.$
9. $\forall a_1, a_2, b_1, b_2.\ (a_1\ R\ b_1$ AND $a_2\ R\ b_2$ AND $b_1 \neq b_2)$ IMPLIES $a_1 \neq a_2.$

(b) Give an example of a relation R that satisfies three of the properties surjection, injection, total, and function (you indicate which) but is not a bijection.

Problem 4.31.
Prove that if relation $R : A \to B$ is a total injection, [≥ 1 out], [≤ 1 in], then

$$R^{-1} \circ R = \mathrm{Id}_A,$$

where Id_A is the identity function on A.

(A simple argument in terms of ”arrows” will do the job.)

Problem 4.32.
Let $R : A \to B$ be a binary relation.

(a) Prove that R is a function iff $R \circ R^{-1} \subseteq \text{Id}_B$.

Write similar containment formulas involving $R^{-1} \circ R$, $R \circ R^{-1}$, Id_a, Id_B equivalent to the assertion that R has each of the following properties. No proof is required.

(b) total.

(c) a surjection.

(d) a injection.

Problem 4.33.
Let $R : A \to B$ and $S : B \to C$ be binary relations such that $S \circ R$ is a bijection and $|A| = 2$.

Give an example of such R, S where neither R nor S is a function.

Hint: Let $|B| = 4$.

Problems for Section 4.5

Practice Problems

Problem 4.34.
Assume $f : A \to B$ is total function, and A is finite. Replace the $\star$ with one of $\leq, =, \geq$ to produce the *strongest* correct version of the following statements:

(a) $|f(A)| \star |B|$.

(b) If f is a surjection, then $|A| \star |B|$.

(c) If f is a surjection, then $|f(A)| \star |B|$.

(d) If f is an injection, then $|f(A)| \star |A|$.

(e) If f is a bijection, then $|A| \star |B|$.

Class Problems

Problem 4.35.
Let $A = \{a_0, a_1, \ldots, a_{n-1}\}$ be a set of size n, and $B = \{b_0, b_1, \ldots, b_{m-1}\}$ a set of size m. Prove that $|A \times B| = mn$ by defining a simple bijection from $A \times B$ to the nonnegative integers from 0 to $mn - 1$.

Problem 4.36.
Let $R : A \to B$ be a binary relation. Use an arrow counting argument to prove the following generalization of the Mapping Rule 1.

Lemma. *If R is a function, and $X \subseteq A$, then*

$$|X| \geq |R(X)|.$$