8 Relations

In which our heroes navigate a sea of many related perils, some of which turn out to be precisely equivalent to each other.

8.1 Why You Might Care

People respond in accordance to how you relate to them.

Nelson Mandela (1918–2013), on choosing reconciliation over vengeance
Interview with *Reader's Digest* (April 2005)

Imagine writing a program to implement a student registration system at a college or university. When a student is registering for classes, you'll need to be answer questions of the form "is Alice eligible to be added to the roster for Price Theory?" to decide whether to allow her to click to add that particular course. To do so, you'll need to know Price Theory's prerequisites: what classes must you have already taken passed before you can take Price Theory? And you'll need corresponding data about Alice's academic history: what classes has Alice already taken? You'd most likely use a database to actually store all of the necessary records, but at its heart the key information is just two sets: a set $prerequisiteOf \subseteq Courses \times Courses$, and a set $passed \subseteq Students \times Courses$. (Depending on the school's rules, you might need some numerical information, too, to ensure both that there's an unfilled seat in the class that Alice can occupy, and that Alice has room in her schedule for another class.) Then Alice is eligible to register for Price Theory only if $\langle \text{Alice}, c\rangle \in passed$ for every course c such that $\langle c, \text{Price Theory}\rangle \in prerequisiteOf$.

In this chapter, we'll explore a generalization of functions, called *relations,* that—like *prerequisiteOf* and *passed*—represent arbitrary subsets of $A \times B$. (In Chapter 2, we saw *functions,* which map each element of some input set A to an element of an output set B. A function is a special kind of relation where each input element is related to one and only one element of the output set. For example, a large retailer might be interested in the relation *purchased*, a subset of $Customers \times Products$; notice that the same customer may have purchased many different products—or one, or none at all—so *purchased* is not a function.)

Relations are the critical foundation of *relational databases,* an utterly widespread modern area of CS, underlying many of the tools we all use regularly. (One classical special-purpose programming language for relational databases is called SQL, for "structured query language"; there are other platforms, too.) A relational database stores a (generally quite large!) collection of structured data. Logically, a database is organized as a collection of *tables,* each of which represents a relation, where each *row* of a table represents an element contained in that relation. Fundamental manipulations of these relations can then be used to answer more sophisticated questions about the underlying data. For example, using standard operations in relational databases (and the relations *prerequisiteOf* and *passed* above), we could compute things like (i) a list of every class c for which you have satisfied all prerequisites of c but have not yet passed c; or (ii) a list of people with whom you've taken at least one class; or (iii) a list of people p with whom you've taken at least one class and where p has also taken at least one class that meets condition (i). (Those are the friends you could ask for help when you take that class.) Or that large retailer might want, for a particular user u, to find the 10 products not purchased by u that were most frequently purchased by other users who share, say, at least half of their purchases with u. All of these queries—though sometimes rather brutally complicated to state in English—can be expressed fairly naturally in the language of relations.

We'll start in Section 8.2 with an introduction to the fundamental definitions relevant to relations. In Section 8.3, we'll look at a few properties—reflexivity, symmetry, and transitivity—that some relations have. Finally, in Section 8.4, we'll look at the special types of relations that result particular combinations of those properties: *equivalence relations* (which divide the world into collections of mutually equivalent items) and *order* relations (which rank everything the world according to the relation, possibly with ties). And, along the way, we'll encounter relational databases, along with regular expressions, algorithmic bias, and applications to topics like asymptotics, voting, and computer graphics.

8.2 Formal Introduction

A man is a bundle of relations, a knot of roots, whose flower and fruitage is the world.

Ralph Waldo Emerson (1803–1882)
"History," *Essays: First Series* (1841)

Informally, a *(binary) relation* describes a pairwise relationship that holds for certain pairs of elements from two sets A and B. As an example, here is one particular relation, expressing the "is a component of" relationship between primary and secondary colors:

$$\{\langle \text{blue}, \text{green}\rangle, \langle \text{blue}, \text{purple}\rangle, \langle \text{red}, \text{orange}\rangle, \langle \text{red}, \text{purple}\rangle, \langle \text{yellow}, \text{green}\rangle, \langle \text{yellow}, \text{orange}\rangle\}$$

In other words, these pairs represent a particular relation on the sets $A = \{\text{red}, \text{yellow}, \text{blue}\}$ and $B = \{\text{green}, \text{purple}, \text{orange}\}$. This description of a relation—a pairwise relationship between some of the elements of two sets A and B—is obviously very general. But let's start by considering a few specific examples, which together begin to show the range of the kinds of properties that relations can represent:

Example 8.1: Satisfaction.
Let $A = \{f : \text{truth assignments for } p \text{ and } q\}$ and $B = \{\varphi : \text{propositions over } p \text{ and } q\}$. One interesting relation between elements of A and B denotes whether a particular truth assignment makes a particular proposition true. (This relation is usually called *satisfies*.) For a proposition φ, a truth assignment f either satisfies φ or it doesn't satisfy φ. For example:

- the truth assignment $[p = \text{T}; q = \text{F}]$ satisfies $p \vee q$ (as do all truth assignments save $[p = \text{F}; q = \text{F}]$);
- the truth assignment $[p = \text{T}; q = \text{F}]$ satisfies $p \wedge \neg q$ (and no other truth assignment does);
- every truth assignment in A satisfies $p \vee \neg p$; and
- no truth assignment in A satisfies $q \wedge \neg q$.

(Thus an element of B might be satisfied by zero, one, or more elements of A. Similarly, an element of A might satisfy many different elements of B.)

Example 8.2: Numbers that are not too different.
We'll say that two real numbers $x \in \mathbb{R}$ and $y \in \mathbb{R}$ are *withinHalf* of each other if $|x - y| \leq 0.5$. Thus *withinHalf* is a relation between pairs of real numbers. For example, $\mathit{withinHalf}(2.781828, 3.0)$ and $\mathit{withinHalf}(3.14159, 3.0)$ and $\mathit{withinHalf}(2.5, 3.0)$ and $\mathit{withinHalf}(2.5, 2.0)$.

Note that $\mathit{withinHalf}(x, x)$ holds for any real number x.

Example 8.3: Being related to.
In keeping with the word "relation," we actually use the phrase "is related to" in English to express one specific binary relation on pairs of people—"being in the same family as" (or "being a (blood) relative of"). For example, we can make the true claim that *Rosemary Clooney is related to George Clooney.*

(A related statement is also true: *George Clooney is related to Rosemary Clooney.* The fact that these two statements convey the same information follows from the fact that the *is related to* relation has a property called *symmetry:* for any x and y, it's the case that x is related to y if and only if y is related to x. Not all relations are symmetric, as we'll see in Section 8.3.)

Some qualitatively different types of relations are already peeking out in these few examples (and more properties of relations will reveal themselves as we go further).

Sometimes the relation contains a finite number of pairs, as in the example of primary and secondary colors; sometimes the relation contains an infinite number of pairs, as in *withinHalf*. Sometimes a particular element x is related to every candidate element, sometimes to none.

Sometimes a relation connects elements from two different sets, as in Example 8.1 (satisfaction, which connected truth assignments to propositions); sometimes it connects two elements from the same set, as in Example 8.3 ("is a (blood) relative of," which connects people to people).

And sometimes the relation has some special properties like *reflexivity,* in which every x is related to x itself (as in *withinHalf*), or *symmetry* (as in "is a (blood) relative of").

8.2.1 The Definition of a Relation, Formalized

Technically, a binary relation is simply a subset of the Cartesian product of two sets:

Definition 8.1: (Binary) relation.
A *(binary) relation on* $A \times B$ is a subset of $A \times B$.

Often we'll be interested in a relation on $A \times A$, where the two sets are the same. If there is no danger of confusion, we may refer to a subset of $A \times A$ as simply a *relation on* A.

Here are a few formal examples of relations:

Example 8.4: A few relations, formally.

- $\{\langle 12,1\rangle, \langle 1,2\rangle, \langle 2,3\rangle, \langle 3,4\rangle, \langle 4,5\rangle, \langle 5,6\rangle, \langle 6,7\rangle, \langle 7,8\rangle, \langle 8,9\rangle, \langle 9,10\rangle, \langle 10,11\rangle, \langle 11,12\rangle\}$ is a relation on $\{1,\ldots,12\}$. (Informally, this relation expresses "is one hour before.")
- $\mid$ ("divides") is a relation on $\mathbb{Z}$, where $\mid$ denotes the set $\{\langle d,n\rangle : n \bmod d = 0\}$.
- $\leq$ is a relation on $\mathbb{R}$, where $\leq$ denotes the set $\{\langle x,y\rangle : x$ is no bigger than $y\}$.

- As a reminder, the *power set* of a set S, denoted $\mathscr{P}(S)$, is the set of all subsets of S. For any set S, then, we can define $\subseteq$ as a relation on $\mathscr{P}(S)$, where $\subseteq$ denotes the set

$$\subseteq \; = \{ \langle A, B \rangle \in \mathscr{P}(S) \times \mathscr{P}(S) : [\forall x \in S : x \in A \Rightarrow x \in B] \} .$$

For the set $S = \{1, 2\}$, for example, the relation $\subseteq$ is

$$\subseteq \; = \left\{ \begin{array}{lllll} \langle \varnothing, \varnothing \rangle, & \langle \varnothing, \{1\} \rangle, & \langle \varnothing, \{2\} \rangle, & \langle \varnothing, \{1,2\} \rangle, & \\ \langle \{1\}, \{1\} \rangle, & \langle \{1\}, \{1,2\} \rangle, & \langle \{2\}, \{2\} \rangle, & \langle \{2\}, \{1,2\} \rangle, & \langle \{1,2\}, \{1,2\} \rangle \end{array} \right\} .$$

- $\{$⟨Ron Rivest, 2002⟩, ⟨Adi Shamir, 2002⟩, ⟨Len Adleman, 2002⟩, ⟨Alan Kay, 2003⟩, ⟨Vint Cerf, 2004⟩, ⟨Robert Kahn, 2004⟩, ⟨Peter Naur, 2005⟩, ⟨Frances Allen, 2006⟩$\}$ is a relation on the set *People* $\times$ $\{2002, 2003, 2004, 2005, 2006\}$, representing the relationship between people and any year in which they won a Turing Award.

Taking it further: Rivest, Shamir, and Adleman won Turing Awards for their work in cryptography; see Section 7.5. Kay was an inventor of the paradigm of object-oriented programming. Cerf and Kahn invented the communication protocols that undergird the Internet. Naur made crucial contributions to the design of programming languages, compilers, and software engineering. Allen made foundational contributions to optimizing compilers and parallel computing.

Example 8.5: Bitstring prefixes.
Let *isPrefix* denote the following relation: for two bitstrings x and y, we have $\langle x, y \rangle \in$ *isPrefix* if and only if the bitstring y starts with precisely the symbols contained in x. (After the bits of x, the bitstring y may contain zero or more additional bits.) For example, 001 is a prefix of $\underline{001}110$ and $\underline{001}$, but 001 is not a prefix of 1001. Write down the relation *isPrefix* on bitstrings of length ≤ 2 explicitly, using set notation. (Write ϵ to denote the empty string.)

Solution.

$$\mathit{isPrefix} = \left\{ \begin{array}{lllllll} \langle \epsilon, \epsilon \rangle, & \langle \epsilon, 0 \rangle, & \langle \epsilon, 1 \rangle, & \langle \epsilon, 00 \rangle, & \langle \epsilon, 01 \rangle, & \langle \epsilon, 10 \rangle, & \langle \epsilon, 11 \rangle, \\ \langle 0, 0 \rangle, & \langle 0, 00 \rangle, & \langle 0, 01 \rangle, & \langle 1, 1 \rangle, & \langle 1, 10 \rangle, & \langle 1, 11 \rangle, & \\ \langle 00, 00 \rangle, & \langle 01, 01 \rangle, & \langle 10, 10 \rangle, & \langle 11, 11 \rangle & & & \end{array} \right\}$$

For some relations—for example, $\mid$ and $\leq$ and $\subseteq$ from Example 8.4—it's traditional to write the symbol for the relation *between* the elements that are being related, using so-called *infix notation.* (So we write $3 \leq 3.5$, rather than $\langle 3, 3.5 \rangle \in \leq$.) In general, for a relation R, we may write either $\langle x, y \rangle \in R$ or $x \, R \, y$, depending on context.

Taking it further: Most programming languages use infix notation in their expressions: that is, they place their operators between their operands, as in `(5 + 3) / 2` in Java or Python or C to denote the value $\frac{5+3}{2}$. But some programming languages, like Postscript (the language commonly used by printers) or the language of Hewlett–Packard calculators, use *postfix* notation, where the operator follows the operands. Other languages, like Scheme, use *prefix* notation, in which the operator comes before the operands. (In Postscript, we would write `5 3 add 2 div`; in Scheme, we'd write `(/ (+ 5 3) 2)`.) While we're all much more accustomed to infix notation, one of the advantages of pre- or postfix notation is that the order of operations is unambiguous: compare the ambiguous `5 + 3 / 2` to its two unambiguous postfix alternatives, namely `5 3 2 div add` and `5 3 add 2 div`.

Taking it further: Recall from Chapter 3 that we defined a *predicate* as a Boolean-valued function—that is, P is a function $P : U \to \{\text{True}, \text{False}\}$ for a set U, called the *universe*. (See Definition 3.20.) For example, we considered the predicate $P_{\text{alphabetical}}(x, y) =$ "string x is alphabetically before string y." Binary predicates—when the universe is a set of pairs $U = A \times B$—are very closely related to binary relations. The main difference is that in Chapter 3 we thought of a binary predicate P as a *function* $P : A \times B \to \{\text{True}, \text{False}\}$, whereas here we're thinking of a relation R on $A \times B$ as a *set* $R \subseteq A \times B$ of ordered pairs. For example, the relation $R_{\text{alphabetical}}$ is the set $\{\langle \texttt{AA}, \texttt{AAH} \rangle, \langle \texttt{AA}, \texttt{AARDVARK} \rangle, \ldots, \langle \texttt{ZYZZYVA}, \texttt{ZYZZYVAS} \rangle\}$. And $P_{\text{alphabetical}}(\texttt{AA}, \texttt{AAH}) = \text{True}$, $P_{\text{alphabetical}}(\texttt{AA}, \texttt{ZYZZYVA}) = \text{True}$, and $P_{\text{alphabetical}}(\texttt{BEFORE}, \texttt{AFTER}) = \text{False}$.

But there's a direct translation between these two worldviews. For a relation $R \subseteq A \times B$, define the predicate P_R as

$$P_R(a, b) = \begin{cases} \text{True} & \text{if } \langle a, b \rangle \in R \\ \text{False} & \text{if } \langle a, b \rangle \notin R. \end{cases}$$

The function P_R is known as the *characteristic function* of the set R: that is, it's the function such that $P_R(x) = \text{True}$ if and only if $x \in R$. ($P_{\text{alphabetical}}$ is the characteristic function of $R_{\text{alphabetical}}$.)

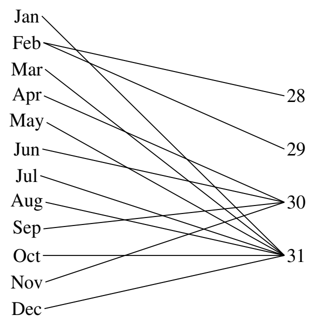
We can also go the other direction, and translate a Boolean-valued binary function into a relation. Given a predicate $P : A \times B \to \{\text{True}, \text{False}\}$, define the relation $R_P = \{\langle a, b \rangle : P(a, b)\}$—that is, define R_P as the set of pairs for which the function P is true. In either case, we have a direct correspondence between (i) the elements of the relation, and (ii) the inputs to the function that make the output true.

Visualizing binary relations

For a relation R on $A \times B$ where both A and B are finite sets, instead of viewing R as a list of pairs, it can be easier to think of R as a two-column table, where each row corresponds to an element $\langle a, b \rangle \in R$. Alternatively, we can visualize relations in a way similar to the way that we visualized functions in Chapter 2: we

Month	Days
Jan	31
Feb	28
Feb	29
Mar	31
Apr	30
May	31
Jun	30
Jul	31
Aug	31
Sep	30
Oct	31
Nov	30
Dec	31

(a) The relation as a table.



(b) The relation, shown visually.

Days	Month
31	Jan
28	Feb
29	Feb
31	Mar
30	Apr
31	May
30	Jun
31	Jul
31	Aug
30	Sep
31	Oct
30	Nov
31	Dec

(c) The inverse of the relation.

Figure 8.1 In (a) and (b), the relation indicating the number of days per month. (Note that Feb is related to *both* 28 and 29.) The inverse of the relation (see Definition 8.2) is shown in (c).

place the elements of A in one column, the elements of B in a second column, and draw a line connecting $a \in A$ to $b \in B$ whenever $\langle a, b\rangle \in R$. Note that when we drew functions using these two-column pictures, every element in the left-hand column had exactly one line leaving it. That's not necessarily true for a relation; elements in the left-hand column could have none, one, or two or more lines leaving them.

Figures 8.1a and 8.1b shows a relation represented in these two ways. (For a relation that's a subset of $A \times A$, the graphical version of this two-column representation is less appropriate because there's really only one kind of element; see Section 8.3 for a different way of visualizing these relations, and see Figure 8.10a for *isPrefix* as a specific example.)

8.2.2 Inverse and Composition of Binary Relations

Because a relation on $A \times B$ is simply a subset of $A \times B$, we can combine relations on $A \times B$ using all the normal set-theoretic operations: if R and S are both relations on $A \times B$, then $R \cup S$, $R \cap S$, and $R - S$ are also relations on $A \times B$, as is the set $\sim R = \{\langle a, b\rangle \in A \times B : \langle a, b\rangle \notin R\}$.

But we can also generate new relations in ways that are specific to relations, rather than being generic set operations. Two of the most common are the *inverse* of a relation (which turns a relation on $A \times B$ into a relation on $B \times A$ by "flipping around" every pair in the relation) and the *composition* of two relations (which turns two relations on $A \times B$ and $B \times C$ into a single relation on $A \times C$, where a and c are related if there's a "two-hop" connection from a to c via some element $b \in B$).

Inverting a relation

Here is the formal definition of the inverse of a relation:

Definition 8.2: Inverse of a relation.
Let R be a relation on $A \times B$. The *inverse* R^{-1} of R is a relation on $B \times A$ defined by $R^{-1} = \{\langle b, a\rangle \in B \times A : \langle a, b\rangle \in R\}$.

Here are a few examples of the inverses of relations:

Example 8.6: Some inverses.

- The inverse of the relation $\leq$ is the relation $\geq$.
- The inverse of the relation $=$ is the relation $=$ itself. (That is, $=$ is its own inverse.)
- The inverse of the month–day relation from Figure 8.1a is shown in Figure 8.1c.
- Define the relation

$$R = \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \langle 1,6\rangle, \langle 2,2\rangle, \langle 2,4\rangle, \langle 2,6\rangle, \langle 3,3\rangle, \langle 3,6\rangle, \langle 4,4\rangle, \langle 5,5\rangle, \langle 6,6\rangle\}.$$

The inverse of R is the relation

$$R^{-1} = \{\langle 2,1\rangle, \langle 3,1\rangle, \langle 4,1\rangle, \langle 5,1\rangle, \langle 6,1\rangle, \langle 2,2\rangle, \langle 4,2\rangle, \langle 6,2\rangle, \langle 3,3\rangle, \langle 6,3\rangle, \langle 4,4\rangle, \langle 5,5\rangle, \langle 6,6\rangle\}.$$

(Note that R is $\{\langle d,n\rangle : d \text{ divides } n\}$, and R^{-1} is $\{\langle n,d\rangle : n \text{ is a multiple of } d\}$.)

Note that, as in the month–day example, the inverse of any relation shown in table form is simply the relation resulting from swapping the two columns of the table.

Composing two relations

The second way of creating a new relation from existing relations is *composition,* which, informally, represents the successive "application" of two relations. Two elements x and y are related under the relation $S \circ R$, denoting the composition of two relations R and S, if there's some intermediate element b that connects x and y under R and S, respectively. (We already saw how to compose *functions,* in Section 2.5, by applying one function immediately after the other. Functions are a special type of relation—see Section 8.2.3—and the composition of functions will similarly be a special case of the composition of relations.) Let's start with an informal example to build some intuition:

Example 8.7: Relation composition, informally.
Consider two relations: *allergicTo* on *People* $\times$ *Ingredients* and *containedIn* on *Ingredients* $\times$ *Entrees*. Then the composition of *allergicTo* and *containedIn* is a relation on *People* $\times$ *Entrees* identifying pairs $\langle p,e\rangle$ for which *entree e contains at least one ingredient to which person p is allergic.*

Here's the formal definition:

Definition 8.3: Composition of two relations.
Let R be a relation on $A \times B$ and let S be a relation on $B \times C$. The *composition* of R and S is a relation on $A \times C$, denoted $S \circ R$, where $\langle a,c\rangle \in S \circ R$ if and only if there exists $b \in B$ such that $\langle a,b\rangle \in R$ and $\langle b,c\rangle \in S$.

Warning! The composition of R and S is, as with functions, denoted $S \circ R$: the function $g \circ f$ *first* applies f and *then* applies g, so $(g \circ f)(x)$ gives the result $g(f(x))$. The order in which the relations are written may initially be confusing.

Perhaps the easiest way to understand the composition of relations is through the picture-based view that we introduced in Figure 8.1b: the relation $S \circ R$ contains pairs of elements that are joined by "two-hop" connections, where the first hop is defined by R and the second hop is defined by S. (See Figure 8.2a.)

Some examples of composing relations

Here are a few examples of the composition of some relations:

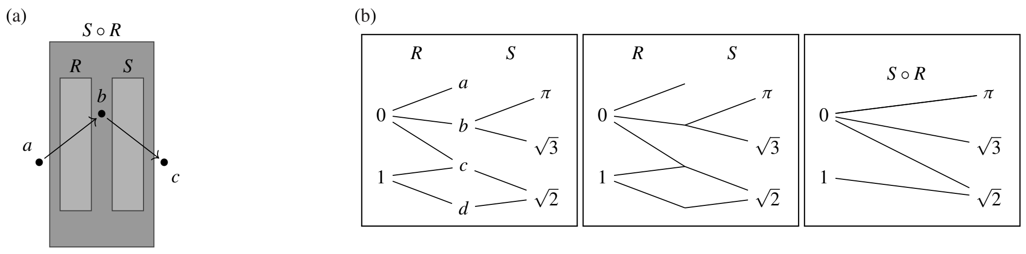


Figure 8.2 The composition of two relations. In (a), the definition: a pair $\langle a, c \rangle$ is in $S \circ R$ when, for some b, both $\langle a, b \rangle \in R$ and also $\langle b, c \rangle \in S$. In (b), a particular example. (See Example 8.8.)

Example 8.8: The composition of two small relations.
Let $R = \{\langle 0, a \rangle, \langle 0, b \rangle, \langle 0, c \rangle, \langle 1, c \rangle, \langle 1, d \rangle\}$ be a relation on $\{0, 1\} \times \{a, b, c, d\}$.
Let $S = \{\langle b, \pi \rangle, \langle b, \sqrt{3} \rangle, \langle c, \sqrt{2} \rangle, \langle d, \sqrt{2} \rangle\}$ be a relation on $\{a, b, c, d\} \times \mathbb{R}$.

Then $S \circ R \subseteq \{0, 1\} \times \mathbb{R}$ is the relation that consists of all pairs $\langle x, z \rangle$ such that there exists an element $y \in \{a, b, c, d\}$ where $\langle x, y \rangle \in R$ and $\langle y, z \rangle \in S$. That is,

$$S \circ R = \left\{ \underbrace{\langle 0, \pi \rangle}_{\text{because of } b}, \underbrace{\langle 0, \sqrt{3} \rangle}_{\text{because of } b}, \underbrace{\langle 0, \sqrt{2} \rangle}_{\text{because of } c}, \underbrace{\langle 1, \sqrt{2} \rangle}_{\text{because of } c \text{ and } d} \right\}.$$

See Figure 8.2b for the visual representation of this composition: because there are "two-hop" paths from 0 to $\{\pi, \sqrt{3}, \sqrt{2}\}$ and from 1 to $\{\sqrt{2}\}$, the relation $S \circ R$ is as described. (Again: the relation $S \circ R$ consists of pairs related by a two-step chain, with the first step under R and the second under S.)

Here's a second example, this time where the relations being composed are more meaningful:

Example 8.9: Relations in the U.S. Senate.
The U.S. Senate has two senators from each state, each of whom is affiliated with zero or one political parties. See Figure 8.3 for two relations: the relation S, between all U.S. states whose names start with the letter "I" and the senators who represented them in the year 1993 (after the 1992 elections, which more than doubled the number of women in the Senate); and the relation T, between senators and their political party. Figure 8.3c shows the composition of S and T, which is a relation between *IStates* and *Parties*, where $\langle \textit{state}, \textit{party} \rangle \in T \circ S$ if one or both of the senators from *state* are affiliated with *party*.

Taking it further: The relation $T \subseteq \textit{Senators} \times \textit{Parties}$, between senators and their political party, is way of representing what's sometimes called an *affiliation network:* we have a collection of individuals (the senators) and groups/organizations (the parties), and information about which individuals are part of which groups (T). Affiliations in other context make sense, too; for example, you might define a relation R on students and university organizations, so that R tells you which people are members of the ballroom dance team or the robotics club and so forth. Some companies have tried to use a relation like R as part of a system to automate their résumé-screening and hiring processes—to some disastrous effects around *algorithmic bias* (in which, unintentionally, the automated system turned out to be making sexist and racist hiring decisions). See p. 8-19 for more.

So far we've considered composing relations on $A \times B$ and $B \times C$ for three distinct sets A, B, and C. But we can also consider a relation $R \subseteq A \times A$, and in this case we can also compose R *with itself.*

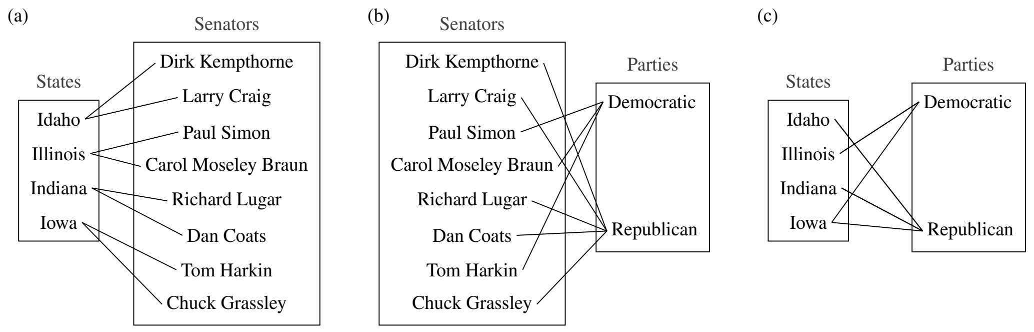


Figure 8.3 Two relations and their composition: (a) the relation $S \subseteq \mathit{IStates} \times \mathit{Senators}$ of each state's senators; (b) the relation $T \subseteq \mathit{Senators} \times \mathit{Parties}$ of each senator's party affiliation; and (c) their composition $T \circ S \subseteq \mathit{IStates} \times \mathit{Parties}$.

Problem-solving tip: Just as you do with a program, always make sure that your mathematical expressions "type check." (For example, just as the Python expression `0.33 * "atomic"` doesn't make sense, the composition $R \circ R$ for the relation $R = \{\langle 1, \mathtt{A} \rangle, \langle 2, \mathtt{B} \rangle\}$ doesn't denote anything useful.)

Example 8.10: Composing a relation with itself.
For each of the following relations R on $\mathbb{Z}^{\geq 1}$, describe the relation $R \circ R$:

1 *successor*, namely the set $\{\langle n, n+1 \rangle : n \in \mathbb{Z}^{\geq 1}\}$.
2 $=$, namely the set $\{\langle n, n \rangle : n \in \mathbb{Z}^{\geq 1}\}$.
3 $\mathit{relativelyPrime} = \{\langle n, m \rangle : \text{GCD}(n, m) = 1\}$, the set of pairs of relatively prime (positive) integers.

Solution. For *successor*: by definition, $\langle x, z \rangle \in \mathit{successor} \circ \mathit{successor}$ if and only if there exists an integer y such that *both* $\langle x, y \rangle \in \mathit{successor}$ *and* $\langle y, z \rangle \in \mathit{successor}$. Thus the only possible y is $y = x + 1$, and the only possible z is $z = y + 1 = x + 2$. Thus $\mathit{successor} \circ \mathit{successor} = \{\langle n, n+2 \rangle : n \in \mathbb{Z}^{\geq 1}\}$.

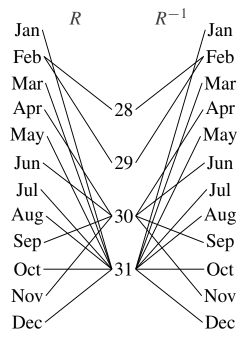
For equality (we'll write *equals* instead of $=$; otherwise the notation becomes indecipherable): by definition, the pair $\langle x, z \rangle$ is in the relation $\mathit{equals} \circ \mathit{equals}$ if and only if there exists an integer y such that $x = y$ and $y = z$. But that's true if and only if $x = z$. That is, $\langle x, z \rangle \in \mathit{equals} \circ \mathit{equals}$ if and only if $\langle x, z \rangle \in \mathit{equals}$. Thus composing *equals* with itself doesn't do anything: $\mathit{equals} \circ \mathit{equals} = \mathit{equals}$.

For relative primality: we must identify all pairs $\langle x, z \rangle \in \mathbb{Z}^{\geq 1} \times \mathbb{Z}^{\geq 1}$ such that there exists an integer y where $\langle x, y \rangle \in \mathit{relativelyPrime}$ *and* $\langle y, z \rangle \in \mathit{relativelyPrime}$. But notice that $y = 1$ is relatively prime to *every* positive integer. Thus, for any $\langle x, z \rangle \in \mathbb{Z}^{\geq 1} \times \mathbb{Z}^{\geq 1}$, we have that $\langle x, 1 \rangle \in \mathit{relativelyPrime}$ and $\langle 1, z \rangle \in \mathit{relativelyPrime}$. Thus $\mathit{relativelyPrime} \circ \mathit{relativelyPrime} = \mathbb{Z}^{\geq 1} \times \mathbb{Z}^{\geq 1}$.

An example of composing a relation with its own inverse

We'll close with one last example of composing relations, this time by taking the composition of a relation R and its inverse R^{-1}:

(a)



(b)

$$\left\{\begin{array}{l}
\langle \text{Jan, Jan}\rangle, \langle \text{Jan, Mar}\rangle, \langle \text{Jan, May}\rangle, \langle \text{Jan, Jul}\rangle, \langle \text{Jan, Aug}\rangle, \langle \text{Jan, Oct}\rangle, \langle \text{Jan, Dec}\rangle,\\
\langle \text{Mar, Jan}\rangle, \langle \text{Mar, Mar}\rangle, \langle \text{Mar, May}\rangle, \langle \text{Mar, Jul}\rangle, \langle \text{Mar, Aug}\rangle, \langle \text{Mar, Oct}\rangle, \langle \text{Mar, Dec}\rangle,\\
\langle \text{May, Jan}\rangle, \langle \text{May, Mar}\rangle, \langle \text{May, May}\rangle, \langle \text{May, Jul}\rangle, \langle \text{May, Aug}\rangle, \langle \text{May, Oct}\rangle, \langle \text{May, Dec}\rangle,\\
\langle \text{Jul, Jan}\rangle, \langle \text{Jul, Mar}\rangle, \langle \text{Jul, May}\rangle, \langle \text{Jul, Jul}\rangle, \langle \text{Jul, Aug}\rangle, \langle \text{Jul, Oct}\rangle, \langle \text{Jul, Dec}\rangle,\\
\langle \text{Oct, Jan}\rangle, \langle \text{Oct, Mar}\rangle, \langle \text{Oct, May}\rangle, \langle \text{Oct, Jul}\rangle, \langle \text{Oct, Aug}\rangle, \langle \text{Oct, Oct}\rangle, \langle \text{Oct, Dec}\rangle,\\
\langle \text{Dec, Jan}\rangle, \langle \text{Dec, Mar}\rangle, \langle \text{Dec, May}\rangle, \langle \text{Dec, Jul}\rangle, \langle \text{Dec, Aug}\rangle, \langle \text{Dec, Oct}\rangle, \langle \text{Dec, Dec}\rangle,\\
\\
\langle \text{Apr, Apr}\rangle, \langle \text{Apr, Jun}\rangle, \langle \text{Apr, Sep}\rangle, \langle \text{Apr, Nov}\rangle,\\
\langle \text{Jun, Apr}\rangle, \langle \text{Jun, Jun}\rangle, \langle \text{Jun, Sep}\rangle, \langle \text{Jun, Nov}\rangle,\\
\langle \text{Sep, Apr}\rangle, \langle \text{Sep, Jun}\rangle, \langle \text{Sep, Sep}\rangle, \langle \text{Sep, Nov}\rangle,\\
\langle \text{Nov, Apr}\rangle, \langle \text{Nov, Jun}\rangle, \langle \text{Nov, Sep}\rangle, \langle \text{Nov, Nov}\rangle,\\
\\
\langle \text{Feb, Feb}\rangle
\end{array}\right\}$$

Figure 8.4 $R^{-1} \circ R$, for the relations R and R^{-1} from Figure 8.1, shown in two ways. In (a), the composition consists of every pair of elements joined by a two-hop path; in (b), the set $R^{-1} \circ R$ is listed explicitly.

Example 8.11: Composing a relation and its inverse.

Let $R \subseteq M \times D$ be the relation between the months and the numbers of days in that month, and let $R^{-1} \subseteq D \times M$ be its inverse. (See Figure 8.1.) What is $R^{-1} \circ R$?

Solution. Because $R \subseteq M \times D$ and $R^{-1} \subseteq D \times M$, we know that $R^{-1} \circ R \subseteq M \times M$. We have to identify

$$\langle x, y\rangle \in M \times M \text{ such that } \exists z \in D : \langle x, z\rangle \in R \text{ and } \langle z, y\rangle \in R^{-1}$$
$$\Leftrightarrow \; \exists z \in D : \langle x, z\rangle \in R \text{ and } \langle y, z\rangle \in R. \qquad \textit{definition of inverse}$$

In other words, we seek pairs of months that are related by R to at least one of the same values. The exhaustive list of pairs in $R^{-1} \circ R$ is shown in Figure 8.4.

Note that $R^{-1} \circ R$ is different from $R \circ R^{-1}$: the latter is the set of numbers that are related by R^{-1} to at least one of the same months, while the former is the set of months that are related by R to at least one of the same numbers. Thus $R \circ R^{-1} = \{\langle 31, 31\rangle, \langle 30, 30\rangle, \langle 29, 29\rangle, \langle 28, 28\rangle, \langle 28, 29\rangle, \langle 29, 28\rangle\}$. (The only distinct numbers related by $R \circ R^{-1}$ are 28 and 29, because of February.)

The relation $R^{-1} \circ R$ from Example 8.11 has a special form: this relation "partitions" the twelve months into three clusters—the 31-day months, the 30-day months, and February—so that any two months in the same cluster are related by $R^{-1} \circ R$, and no two months in different clusters are related by $R^{-1} \circ R$. (See Figure 8.10b.) A relation with this structure, where elements are partitioned into clusters (and two elements are related if and only if they're in the same cluster) is called an *equivalence relation*; see Section 8.4.1.

8.2.3 Functions as Relations

Back in Chapter 2, we defined a *function* as something that maps each element of the set of legal inputs (the *domain*) to an element of the set of legal outputs (the *range*):

Definition 2.46 (functions): Let A and B be sets. A *function f from A to B*, written $f : A \to B$, assigns to each input value $a \in A$ a unique output value $b \in B$; the unique value b assigned to a is denoted by $f(a)$. We sometimes say that *f maps a to $f(a)$*.

While we've begun this chapter defining relations as a completely different kind of thing from functions, actually functions are just a special type of relation. For example, the "one hour later than" relation $\{\langle 12,1\rangle, \langle 1,2\rangle, \ldots, \langle 10,11\rangle, \langle 11,12\rangle\}$ from Example 8.4 really *is* a function $f : \{1,\ldots,12\} \to \{1,\ldots,12\}$, where we could write f more compactly as $f(x) = (x \bmod 12) + 1$.

In general, to think of a function $f : A \to B$ as a relation, we will view f as defining the set of ordered pairs $\langle x, f(x)\rangle$ for each $x \in A$, rather than as a mapping:

Definition 8.4: Functions, viewed as relations.
Let A and B be sets. A *function f from A to B*, written $f : A \to B$, is a relation on $A \times B$ with the additional property that, for every $a \in A$, there exists one and only one element $b \in B$ such that $\langle a, b\rangle \in f$.

That is, we view the function $f : A \to B$ as the set $F = \{\langle x, f(x)\rangle : x \in A\}$, which is a subset of $A \times B$. The restriction of the definition requires that F has a unique output defined for every input: there cannot be two distinct pairs $\langle x, y\rangle$ and $\langle x, y'\rangle$ in F, and furthermore there cannot be any x for which there's no $\langle x, \bullet\rangle$ in F.

Example 8.12: A function as a relation.
(Write $\mathbb{Z}_{11}$ to denote $\{0,1,2,\ldots,10\}$, as in Chapter 7.) The function $f : \mathbb{Z}_{11} \to \mathbb{Z}_{11}$ defined as $f(x) = x^2 \bmod 11$ can be written as

$$\{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,4\rangle, \langle 3,9\rangle, \langle 4,5\rangle, \langle 5,3\rangle, \langle 6,3\rangle, \langle 7,5\rangle, \langle 8,9\rangle, \langle 9,4\rangle, \langle 10,1\rangle\}.$$

Observe that f^{-1}, the inverse of f, is *not* a function—for example, the pairs $\langle 5,4\rangle$ and $\langle 5,7\rangle$ are both in f^{-1}, and there is no element $\langle 2, \bullet\rangle \in f^{-1}$. But f^{-1} *is* still a relation.

Example 8.13: Composing functions.
Suppose that $f \subseteq A \times B$ and $g \subseteq B \times C$ are functions (in the sense of Definition 8.4). Prove that the relation $g \circ f$ is a function from A to C.

Solution. By definition, the composition of the relations f and g is

$$g \circ f = \{\langle x, z\rangle : \text{there exists } y \text{ such that } \langle x, y\rangle \in f \text{ and } \langle y, z\rangle \in g\}.$$

Consider any $x \in A$. Because f is a function, there exists one and only one y^* such that $\langle x, y^*\rangle \in f$. Furthermore, because g is a function, for this particular y^* there is a unique z with $\langle y^*, z\rangle \in g$. Thus there exists one and only one z such that $\langle x, z\rangle \in g \circ f$. By definition, then, the relation $g \circ f$ is a function.

Under this functions-as-relations view, the definitions of the inverse and composition of functions—Definitions 2.50 and 2.54—precisely line up with the definitions of the inverse and composition of relations from this section. Furthermore, if a function is just a special type of relation, then the special types of functions that we defined in Chapter 2—one-to-one and onto functions—are just further restrictions on relations. Under the relation-based view of functions, the function $f \subseteq A \times B$ is called *one-to-one* if, for every $b \in B$, there exists at most one element $a \in A$ such that $\langle a, b \rangle \in f$. The function $f \subseteq A \times B$ is called *onto* if, for every $b \in B$, there exists at least one element $a \in A$ such that $\langle a, b \rangle \in f$.

If $f \subseteq A \times B$ is a function, then the inverse f^{-1} of f—that is, the set $f^{-1} = \{\langle b, a \rangle : \langle a, b \rangle \in f\}$—is guaranteed to be a *relation* on $B \times A$. But f^{-1} is a *function* from B to A if and only if f is both one-to-one and onto. In Exercises 8.39–8.44, you'll explore some other properties of the composition of functions/relations.

8.2.4 n-ary Relations

The relations that we've explored so far have all expressed relationships between *two* elements. But some interesting properties might involve more than two entities; for example, you might assemble all of your friends' birthdays as a collection of *triples* of the form $\langle name, birthdate, birthyear \rangle$. Or we might consider a relation on integers of the form $\langle a, b, k \rangle$ where $a \equiv_k b$. A relation involving tuples with n components, called an *n-ary relation,* is a natural generalization of a (binary) relation:

Definition 8.5: n-ary relation.
An *n-ary relation* on the set $A_1 \times A_2 \times \cdots \times A_n$ is a subset of $A_1 \times A_2 \times \cdots \times A_n$. If there is no danger of confusion, we may refer to a subset of A^n as an n-ary relation on A.

(We generally refer to 2-ary relations as *binary relations* and 3-ary relations as *ternary relations*.) Here are a few examples:

Example 8.14: Summing to 8.
Define $sumsTo8$ as a ternary relation on the set $\{0, 1, 2, 3, 4\}$, where

$$sumsTo8 = \{\langle a, b, c \rangle \in \{0,1,2,3,4\} \times \{0,1,2,3,4\} \times \{0,1,2,3,4\} : a + b + c = 8\}.$$

Color	R	G	B
Black	0	0	0
Blue	0	0	255
Cyan	0	255	255
Gray	128	128	128
Green	0	128	0
Lime	0	255	0
Magenta	255	0	255
Maroon	128	0	0
Navy	0	0	128
Olive	128	128	0
Purple	128	0	128
Red	255	0	0
Teal	0	128	128
White	255	255	255
Yellow	255	255	0

Figure 8.5 The 4-ary relation containing the full set of RGB colors with component values that are all drawn from either $\{0, 128\}$ or $\{0, 255\}$.

Then *sumsTo*8 is

$$\left\{\begin{array}{l}\langle 0,4,4\rangle, \langle 1,3,4\rangle, \langle 1,4,3\rangle, \langle 2,2,4\rangle, \langle 2,3,3\rangle, \langle 2,4,2\rangle, \\ \langle 3,1,4\rangle, \langle 3,2,3\rangle, \langle 3,3,2\rangle, \langle 3,4,1\rangle, \langle 4,0,4\rangle, \langle 4,1,3\rangle, \langle 4,2,2\rangle, \langle 4,3,1\rangle, \langle 4,4,0\rangle\end{array}\right\}.$$

Example 8.15: Betweenness.

Define the set $B = \{\langle x,y,z\rangle \in \mathbb{R}^3 : x \leq y \leq z \text{ or } x \geq y \geq z\}$. Then B is a ternary relation on $\mathbb{R}$ that expresses "betweenness"—that is, the triple $\langle x,y,z\rangle \in B$ if x, y, and z are in a consistent order (either ascending or descending).

For example, we have $\langle -1,0,1\rangle \in B$ and $\langle 6,5,4\rangle \in B$, because $-1 \leq 0 \leq 1$ and $6 \geq 5 \geq 4$. But $\langle -7,8,-9\rangle \notin B$, because these three numbers are neither in ascending order (because $8 \not\leq -9$) nor descending order (because $-7 \not\geq 8$).

Example 8.16: RGB colors.

A 4-ary relation on *Names* $\times \{0,1,\ldots,255\} \times \{0,1,\ldots,255\} \times \{0,1,\ldots,255\}$ is shown in Figure 8.5: a collection of colors, each with its official name in HTML/CSS and its red, green, and blue components—all of which are elements of $\{0,1,\ldots,255\}$. (*HTML (hypertext markup language)* and *CSS (cascading style sheet)* are languages used to express the format, style, and layout of web pages.)

Taking it further: *Databases*—systems for storing and accessing collections of structured data—are a widespread modern application of computer science. Databases store student records for registrars, account information for financial institutions, and records of who liked whose posts on Facebook; in short, virtually every industrial system that has complex data with nontrivial relationships among data elements is stored in a database. More specifically, a *relational database* stores information about a collection of entities and relationships among those entities: fundamentally, a relational database is a collection of n-ary relations, which can then be manipulated and queried in various ways. Designing databases well affects both how easy it is for a user to pose the questions that they wish to ask about the data, *and* how efficiently answers to those questions can be computed. See p. 8-17 for more on relational databases and how they connect with the types of relations that we've discussed so far.

Expressing *n*-ary relations as a collection of binary relations

Non-binary relations, like those in the last few examples, represent complex interactions among more than two entities. For example, the "betweenness" relation

$$B = \{\langle x,y,z\rangle \in \mathbb{R}^3 : x \leq y \leq z \text{ or } x \geq y \geq z\}$$

from Example 8.15 fundamentally expresses a relationship regarding triples of numbers: for any three real numbers x, y, and z, there are triples $\langle x,y,\bullet\rangle \in B$ and $\langle \bullet,y,z\rangle \in B$ and $\langle x,\bullet,z\rangle \in B$—but whether $\langle x,y,z\rangle$ itself is in the relation B genuinely depends on how all three numbers relate to each other. Similarly, the *sumsTo*8 relation from Example 8.14 is a genuinely three-way relationship among elements—not something that can be directly reduced to a pair of pairwise relationships. But we *can* represent an n-ary relation

R by a collection of binary relations, if we're a little creative in defining the sets that are being related. (Decomposing n-ary relations into multiple binary relations may be helpful if we store this type of data in a database; there may be advantages of clarity and efficiency in this view of an n-ary relation.)

This idea is perhaps easiest to see for the colors from Example 8.16: because each color name appears once and only once in the table, we can treat the name as unique "key" that allows us to treat the 4-ary relation as three separate binary relations, corresponding to the red, green, and blue components of the colors. (See Figure 8.6a.) But how would we represent an n-ary relation like the ternary *sumsTo*8 using multiple binary relations? (Recall the relation

$$sumsTo8 = \left\{ \begin{array}{l} \langle 0,4,4\rangle, \langle 1,3,4\rangle, \langle 1,4,3\rangle, \langle 2,2,4\rangle, \langle 2,3,3\rangle, \langle 2,4,2\rangle, \\ \langle 3,1,4\rangle, \langle 3,2,3\rangle, \langle 3,3,2\rangle, \langle 3,4,1\rangle, \langle 4,0,4\rangle, \langle 4,1,3\rangle, \langle 4,2,2\rangle, \langle 4,3,1\rangle, \langle 4,4,0\rangle \end{array} \right\}$$

from Example 8.14.) One idea is to introduce a new set of fake "entities" that correspond to each of the tuples in *sumsTo*8, and then build binary relations between each component and this set of entities. For example, define the set

$$E = \{044, 134, 143, 224, 233, 242, 314, 323, 332, 341, 404, 413, 422, 431, 440\}\,,$$

and then define the three binary relations *first*, *second*, and *third* shown in Figure 8.6b. Now $\langle a,b,c\rangle \in$ *sumsTo*8 if and only if there exists an $e \in E$ such that $\langle e,a\rangle \in$ *first*, $\langle e,b\rangle \in$ *second*, and $\langle e,c\rangle \in$ *third*. (See Exercise 8.45 for a way to think of betweenness using binary relations.)

R	
Black	0
Blue	0
Cyan	0
Gray	128
Green	0
⋮	

G	
Black	0
Blue	0
Cyan	255
Gray	128
Green	128
⋮	

B	
Black	0
Blue	255
Cyan	255
Gray	128
Green	0
⋮	

(a) The colors from the 4-ary relation in Example 8.16, represented as three binary relations.

first	
044	0
134	1
143	1
224	2
233	2
⋮	

second	
044	4
134	3
143	4
224	2
233	3
⋮	

third	
044	4
134	4
143	3
224	4
233	3
⋮	

(b) The relation *sumsTo*8, as three binary relations.

Figure 8.6 Representing n-ary relations as a collection of binary relations.

Computer Science Connections

Relational Databases

A *database* is a (generally large!) collection of structured data. A user can both "query" the database (asking questions about existing entries and edit it (adding or updating existing entries). The bulk of modern attention to databases focuses on *relational databases,* based explicitly on the types of relations explored in this chapter. (Until a massively influential early paper by Ted Codd (1923–2003) [30]—he later would win a Turing Award for this work—database systems were generally based on rigid top-down organization of the data.)

In a relational database, the fundamental unit of storage is the *table,* which represents an n-ary relation $R \subseteq A_1 \times A_2 \times \cdots \times A_n$. A table consists of a collection of *columns,* each of which represents a component of R; the columns are labeled with the name of the corresponding component so you can refer to columns by name rather than just by index. The *rows* of the table correspond to elements of the relation: that is, each row is a value $\langle a_1, a_2, \ldots, a_n \rangle$ that's in R. Figure 8.7 shows an example table of this form, echoing Example 8.16.

name	*red*	*green*	*blue*
Green	0	128	0
Lime	0	255	0
Magenta	255	0	255
Maroon	128	0	0
Navy	0	0	128
Olive	128	128	0
Purple	128	0	128
Red	255	0	0
Teal	0	128	128
White	255	255	255
Yellow	255	255	0

```
SELECT name, red
FROM colors
WHERE green > blue;
```

name	*red*
Green	0
Lime	0
Olive	128
Yellow	255

Figure 8.7 Some RGB colors; and the SQL code and output resulting from selecting colors with *green* $>$ *blue* and projecting to *name*, *red*.

Thus a relational database is at its essence a collection of n-ary relations. A common way to interact with this sort of database is with a special-purpose programming language, often the language *SQL.* ("SQL" is short for *Structured Query Language*; it's pronounced either like "sequel" or by spelling out the letters [to rhyme with "Bless you, Mel!"].) Operations on relational databases are based on three fundamental operations on n-ary relations. The first two basic operations either choose some of the rows or some of the columns from an n-ary relation $R \subseteq A_1 \times \cdots \times A_n$:

- *select:* for a predicate φ on $A_1 \times \cdots \times A_n$, we can *select* those elements of R that satisfy φ.
- *project:* we can *project* $R \subseteq A_1 \times \cdots \times A_n$ into a smaller set of columns by deleting some A_is.

For example, we can select colors with blue component equal to zero, or project the colors relation down to just red and blue values. See Figure 8.7 for an example. (In SQL, select and project operations are done with unified syntax.)

The third key operation in relational databases, called *join,* corresponds closely to the composition of relations. In a join, we combine two relations by insisting that an identified shared column of the two relations matches. Unlike with the

S

state	*senator*
IA	Grassley
IA	Harkin
ID	Craig
ID	Kempthorne
IL	Moseley Braun
IL	Simon
IN	Coats
IN	Lugar

T

senator	*party*
Coats	R
Craig	R
Grassley	R
Harkin	D
Kempthorne	R
Lugar	R
Moseley Braun	D
Simon	D

```
SELECT * FROM T
INNER JOIN S
  ON T.senator = S.senator;
```

senator	*party*	*state*
Coats	R	IN
Craig	R	ID
Grassley	R	IA
	⋮	

Figure 8.8 Joining S and T from Figure 8.3.

composition of relations, we *continue to include that matching column* in the resulting table. For two binary relations $X \subseteq S \times T$ and $Y \subseteq T \times U$, the *join* of X and Y is a *ternary* relation on $S \times T \times U$, defined as $X \bowtie Y = \{\langle a, c, b\rangle \in S \times T \times U : \langle a, c\rangle \in X \text{ and } \langle c, b\rangle \in Y\}$. In SQL syntax, this operation is denoted by `INNER JOIN`; see Figure 8.8.

(The description here barely scratches the surface of relational databases—there's a full course's worth of material on databases (and then some!) that we've left out, including how these operations are implemented and how to design databases to support efficient operations. For more, see a good book on databases, like [117].)

Computer Science Connections

Automating Decisions, Facial Recognition, and Algorithmic Bias

Imagine that you been put in charge of hiring new university graduates as software engineers for the company where you work. The process of screening résumés can be tedious, and it's potentially prone to whatever implicit biases (about, say, age, or gender, or race, or degree-granting university) you may happen to harbor. So you might think about automating the process, to ease your workload and try to avoid human fallibilities. Here's one natural way to do it: first, you apply some kind of threshold test on applicants' academic qualifications, and, second, you look at characteristics of applicants, and try to infer the likelihood of success at the job for someone with those particular characteristics. There are a lot of characteristics that you might choose to use in this kind of analysis, but one reasonable choice is to look at university organizations that your applicants have participated in: athletic teams, musical groups, hackathons, whatever. So, you might look at the participation relation $R \subseteq \mathit{Applicants} \times \mathit{Clubs}$, and then, to find the most promising applicants in the pool, you might calculate the fraction f_c the fraction of your currently employed software engineers who were in each club c. Then you could score an applicant a by $\sum_{c \in \mathit{Clubs}: \langle a,c \rangle} f_c$ (or something similar but more sophisticated), and interview the applicants in descending order of their score.

Images from official U.S. House of Representatives portraits

Figure 8.9 New York State's 2021 U.S. House of Representatives delegation, from District 1 (top left) to District 27 (bottom right).

The automated résumé-screening system described above is very similar to a system that Amazon tried to build and deploy in 2018—which they quickly scrapped after discovering exactly how it was performing. More specifically, the automated system ended up being brutally sexist: for example, the word "women" on a résumé, as in "women's fencing team captain," was causing the score of that résumé to be massively decreased. (See [35] for the Reuters story that broke the news.) The basic reason that the system was so badly biased because of a particular overlooked piece of logic in the system. *This system is a fancy way of reproducing the existing biases in the workforce* because it tries to predict who will be a good future software engineer *by comparing characteristics to the good current software engineers* (who were hired by the previous generations of human screeners with their own biases).

Algorithmic bias refers to automated systems making discriminatory decisions, typically by reproducing existing biases in their training data. Another prominent example, among too many, is an algorithmic system that's been deployed to decide whether to grant parole [7]. In it, for instance, one of inputs to the probabilistic model of recidivism was the number of previous arrests for the individual—but differential policing can lead to more arrests in some neighborhoods (correlated with race and socioeconomics).

There are other forms of algorithmic bias, too. It's possible to build a system that looks like it does an excellent job at some task—and yet is still profoundly biased, in that its performance is much better when, say, the input corresponds to a white man than when the input corresponds to a person with any other demographic characteristics.

For example, Joy Buolamwini and Timnit Gebru [24] showed a severe asymmetry of accuracy rates across skin tones in modern commercial image-processing systems presented with faces, like those in Figure 8.9, as input. (These systems can try to detect faces, or recognize faces, or predict the gender of faces, etc.—and the performance of all of them was systematically worse when presented with images of individuals with darker skin tones.) For more on the kinds of flaws arising in (computational and noncomputational) research resulting from building models based only on men in training data, see Caroline Criado Perez's *Invisible Women* [99]. And for more of the burgeoning research by computer scientists on understanding and mitigating these kind of algorithmic bias issues, see the proceedings of the ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT), `facctconference.org`.

EXERCISES

Here are a few English-language descriptions of relations on a particular set. For each, write out (by exhaustive enumeration) the full set of pairs in the relation, as we did in Examples 8.4 and 8.5. (Hint: It's easy to miss an element of these relations if you solve these problems by hand. Consider writing a small program to enumerate all pairs in these relations.)

8.1 *divides*, written $|$, on $\{1, 2, \ldots, 8\}$, where $\langle d, n \rangle \in |$ if and only if $n \bmod d = 0$.

8.2 *subset*, written $\subset$, on $\mathscr{P}(\{1, 2, 3\})$, where $\langle S, T \rangle \in \subset$ if and only if $S \neq T$ and $\forall x : x \in S \Rightarrow x \in T$.

8.3 *isProperPrefix* on bitstrings of length ≤ 3. A string x is a proper prefix of a string y if x starts with precisely the symbols of y, followed by one or more other symbols. (See Example 8.5, but note that we are looking for *proper* prefixes here. The string x is prefix, but not a proper prefix, of itself.)

8.4 *isProperSubstring* on bitstrings of length ≤ 3. For two strings x and y, we say that x is a *substring* of y if all of x appears consecutively somewhere in y. And x is a *proper* substring of y if x is a substring of y but $x \neq y$.

8.5 *isProperSubsequence* on bitstrings of length ≤ 3. A string x is a *subsequence* of y if the symbols of x appear in order, but not necessarily consecutively, in y. (For example, 001 is a substring of $1\underline{001}$ but not of 0101. But 001 is a subsequence of $1\underline{001}$ and also of $\underline{0}1\underline{01}$.) Again, x is *proper* subsequence of y if x is a subsequence of y but $x \neq y$.

8.6 *isAnagram* on bitstrings of length ≤ 3. A string x is an anagram of a string y if x contains exactly the same symbols as y (with the same number of copies of each symbol), but not necessarily in the same order.

Let $\subseteq$ and $\subset$ denote the subset and proper subset relations on $\mathscr{P}(\mathbb{Z})$. (That is, we have $\langle A, B \rangle \in \subset$ if $A \subseteq B$ but $A \neq B$.) What relation is represented by each of the following?

8.7 $\subseteq \cup \subset$

8.8 $\subseteq - \subset$

8.9 $\subset - \subseteq$

8.10 $\subset \cap \subseteq$

8.11 $\sim\subset$

Define $R = \{\langle 2, 2 \rangle, \langle 5, 1 \rangle, \langle 2, 3 \rangle, \langle 5, 2 \rangle, \langle 2, 1 \rangle\}$ *and* $S = \{\langle 3, 4 \rangle, \langle 5, 3 \rangle, \langle 6, 6 \rangle, \langle 1, 4 \rangle, \langle 4, 3 \rangle\}$ *as two relations on the set* $\{1, 2, 3, 4, 5, 6\}$. *What pairs are in the following relations?*

8.12 R^{-1}

8.13 S^{-1}

8.14 $R \circ R$

8.15 $R \circ S$

8.16 $S \circ R$

8.17 $R \circ S^{-1}$

8.18 $S \circ R^{-1}$

8.19 $S^{-1} \circ S$

Five so-called mother sauces *of French cooking were codified by the chef Auguste Escoffier in the early 20th century. (Many other sauces—"daughter" or "secondary" sauces—used in French cooking are derived from these basic recipes.) They are:*

- Sauce Béchamel *is made of milk, butter, and flour.*
- Sauce Espagnole *is made of stock, butter, and flour.*
- Sauce Hollandaise *is made of egg, butter, and lemon juice.*
- Sauce Velouté *is made of stock, butter, and flour.*
- Sauce Tomate *is made of tomatoes, butter, and flour.*

8.20 Write down the "is an ingredient of" relation on *Ingredients* $\times$ *Sauces* using the tabular representation of relations introduced in Figure 8.1.

8.21 Writing R to denote the relation that you enumerated in Exercise 8.20, what is $R \circ R^{-1}$? Give both a list of elements and an English-language description of what $R \circ R^{-1}$ represents.

8.22 For R from Exercise 8.20, what is $R^{-1} \circ R$? Again, give both a list of elements and a description of the meaning.

Suppose that a Registrar's office has computed the following relations:

$$\text{taughtIn} \subseteq \text{Classes} \times \text{Rooms} \qquad \text{taking} \subseteq \text{Students} \times \text{Classes} \qquad \text{at} \subseteq \text{Classes} \times \text{Times}.$$

For the following exercises, express the given additional relation using taughtIn, taking, *and* at, *plus relation composition and/or inversion (and no other tools).*

8.23 $R \subseteq \mathit{Students} \times \mathit{Times}$, where $\langle s, t \rangle \in R$ indicates that student s is taking a class at time t.

8.24 $R \subseteq \mathit{Rooms} \times \mathit{Times}$, where $\langle r, t \rangle \in R$ indicates that there is a class in room r at time t.

8.25 $R \subseteq \mathit{Students} \times \mathit{Students}$, where $\langle s, s' \rangle \in R$ indicates that students s and s' are taking at least one class in common.

8.26 $R \subseteq \mathit{Students} \times \mathit{Students}$, where $\langle s, s' \rangle \in R$ indicates that there's at least one time when s and s' are both taking a class (but not necessarily the same class).

Let parent $\subseteq$ People $\times$ People *denote the relation* $\{\langle p, c \rangle : p \text{ is a parent of } c\}$. *(For the sake of simplicity over realism, assume that there are no divorces, remarriages, widows, widowers, adoptions, single parents, etc. That is, you should assume that each child has exactly two parents, and any two children who share one parent share both parents.) What familial relationships are represented by the following relations?*

8.27 $\mathit{parent} \circ \mathit{parent}$

8.28 $(\mathit{parent}^{-1}) \circ (\mathit{parent}^{-1})$

8.29 $\mathit{parent} \circ (\mathit{parent}^{-1})$

8.30 $(\mathit{parent}^{-1}) \circ \mathit{parent}$

8.31 $\mathit{parent} \circ \mathit{parent} \circ (\mathit{parent}^{-1}) \circ (\mathit{parent}^{-1})$

8.32 $\mathit{parent} \circ (\mathit{parent}^{-1}) \circ \mathit{parent} \circ (\mathit{parent}^{-1})$

8.33 Suppose that the relations $R \subseteq \mathbb{Z} \times \mathbb{Z}$ and $S \subseteq \mathbb{Z} \times \mathbb{Z}$ contain, respectively, n pairs and m pairs of elements. In terms of n and m, what's the largest possible size of $R \circ S$? The smallest?

8.34 For arbitrary relations R, S, and T, prove that $R \circ (S \circ T) = (R \circ S) \circ T$.

8.35 For arbitrary relations R and S, prove that $(R \circ S)^{-1} = (S^{-1} \circ R^{-1})$.

8.36 Let R be any relation on $A \times B$. Prove or disprove: $\langle x, x \rangle \in R \circ R^{-1}$ for every $x \in A$.

8.37 What set is represented by the relation $\leq \circ \geq$, where $\leq$ and $\geq$ are relations on $\mathbb{R}$?

8.38 What set is represented by the relation $\mathit{successor} \circ \mathit{predecessor}$, for the relations $\mathit{successor} = \{\langle n, n+1 \rangle : n \in \mathbb{Z}\}$ and $\mathit{predecessor} = \{\langle n, n-1 \rangle : n \in \mathbb{Z}\}$?

Suppose that $R \subseteq A \times B$ and $T \subseteq B \times C$ are relations. The first few exercises below (Exercises 8.39–8.41) ask you to prove that some properties of R and S translate into properties of $T \circ R$. The next few exercises (Exercises 8.42–8.44) ask you to address the converse of these results. Supposing that $T \circ R$ has the listed property, can you infer that both relations R and T have the same property? Only R? Only T? Neither? Prove your answers.

8.39 Prove that, if R and T are both functions, then $T \circ R$ is a function too.

8.40 Prove that, if R and T are both one-to-one functions, then $T \circ R$ is one-to-one too.

8.41 Prove that, if R and T are both onto functions, then $T \circ R$ is onto too.

8.42 Assume that $T \circ R$ is a function. Must T be a function? R? Both?

8.43 Assume that $T \circ R$ is a one-to-one function and that R and T are both functions. Must T be one-to-one? R? Both?

8.44 Assume that $T \circ R$ is an onto function and that R and T are both functions. Must T be onto? R? Both?

On p. 8-17, we introduced three operations on relations that are used frequently in relational databases:

- select(R, P)*: choose a subset of an n-ary relation R, according to some condition P. (A "condition" P assigns true or false to each element of the universe.)*
- project(R, K)*: turn an n-ary relation R into an k-ary relation for some $k \leq n$, by eliminating those columns of R that aren't in K. (The set $K \subseteq \{1, 2, \ldots, n\}$ identifies which columns of R to keep.)*
- join(R, S)*: combine two binary relations $R \subseteq A \times B$ and $S \subseteq B \times C$ into a single ternary relation containing all triples $\langle a, b, c \rangle$ where $\langle a, b \rangle \in R$ and $\langle b, c \rangle \in S$.*

For example, let $R = \{\langle 1,2,3\rangle, \langle 4,5,6\rangle\}$*, let* $S = \{\langle 6,7\rangle, \langle 6,8\rangle\}$*, and let* $T = \{\langle 7,9\rangle, \langle 7,10\rangle\}$*. Then*

- $\text{select}(R, \text{xzEven}) = \{\langle 4,5,6\rangle\}$ *for* $\text{xzEven}(x,y,z) = (2 \mid x) \wedge (2 \mid z)$.
- $\text{project}(\text{R}, \{1,2\}) = \{\langle 1,2\rangle, \langle 4,5\rangle\}$ *and* $\text{project}(\text{R}, \{1,3\}) = \{\langle 1,3\rangle, \langle 4,6\rangle\}$.
- $\text{join}(S,T) = \{\langle 6,7,9\rangle, \langle 6,7,10\rangle\}$.

Solve the following using the relation operators $^{-1}$ *(inverse),* $\circ$ *(composition), select, project, and join:*

8.45 Recall from Example 8.15 the ternary "betweenness" relation $B = \big\{\langle x,y,z\rangle \in \mathbb{R}^3 : x \leq y \leq z \text{ or } x \geq y \geq z\big\}$. Show how to construct B using only $\leq$, the relation operators ($^{-1}$, $\circ$, join, select, project), and standard set-theoretic operations ($\cup$, $\cap$, $\sim$, $-$).

Figure 8.5 contains a reminder of the 4*-ary relation C that lists several colors and their red, green, and blue components. Using this relation C and select/project/join, write a set that corresponds to the following:*

8.46 the names of all colors that have red component 0.

8.47 the names of all pairs of colors whose amount of blue is the same.

8.48 the names of all colors that are more blue than red.

Let X denote the set of color names from Figure 8.5. Define three relations Red, Green*, and* Blue *on* $X \times \{0,1,\ldots,255\}$ *such that* $\langle x,r,g,b\rangle \in C$ *if and only if* $\langle x,r\rangle \in \text{Red}$ *and* $\langle x,g\rangle \in \text{Green}$ *and* $\langle x,b\rangle \in \text{Blue}$*. (In other words,* $\text{Red} = \text{project}(C, \{1,2\})$ *and* $\text{Green} = \text{project}(C, \{1,3\})$ *and* $\text{Blue} = \text{project}(C, \{1,4\})$*.)*

8.49 Repeat Exercise 8.47 using only $^{-1}$, $\circ$, and the relations *Red*, *Green*, *Blue*, $\leq$, and $=$.

8.50 Repeat Exercise 8.48 using only $^{-1}$, $\circ$, and the relations *Red*, *Green*, *Blue*, $\leq$, and $=$. Or, at least, compute the set of $\langle x,x\rangle$ such that x is the name of a color that's more blue than red. (You may construct a relation R on colors, and then take $R \cap =$.)

8.3 Properties of Relations: Reflexivity, Symmetry, and Transitivity

There are two ways of spreading light; to be
The candle or the mirror that reflects it.

Edith Wharton (1862–1937)
"Vesalius in Zante (1564)" (1902)

Let $R \subseteq A \times A$ be a relation on a single set A (as in the *successor* or $\leq$ relations on $\mathbb{Z}$, or the *is a (blood) relative of* relation on people). We've seen a two-column approach to visualizing a relation $R \subseteq A \times B$, but this layout is misleading when the sets A and B are identical. (Weirdly, we'd have to draw each element twice, in both the A column and the B column.) Instead, it will be more convenient to visualize a relation $R \subseteq A \times A$ without differentiated columns, using a *directed graph*: we simply write down each element of A, and draw an arrow from a_1 to a_2 for every pair $\langle a_1, a_2 \rangle \in R$. (See Chapter 11 for much more on directed graphs.) A few small examples are shown in Figure 8.10.

This directed-graph visualization of relations will provide a useful way of thinking intuitively about relations in general—and about some specific types of relations in particular. There are several important structural properties that some relations on A have (and that some relations do not), and we'll explore these properties throughout this section. We'll consider three basic categories of properties:

reflexivity: whether elements are related to themselves. Is an element x necessarily related to x itself?

symmetry: whether order matters in the relation. If x and y are related, are y and x necessarily related too?

transitivity: whether chains of related pairs are themselves related. If x and y are related and y and z are related, are x and z necessarily related too?

These properties turn out to characterize several important types of relations—for example, some relations divide A into clusters of "equivalent" elements (as in Figure 8.10b), while other relations "order" A in some

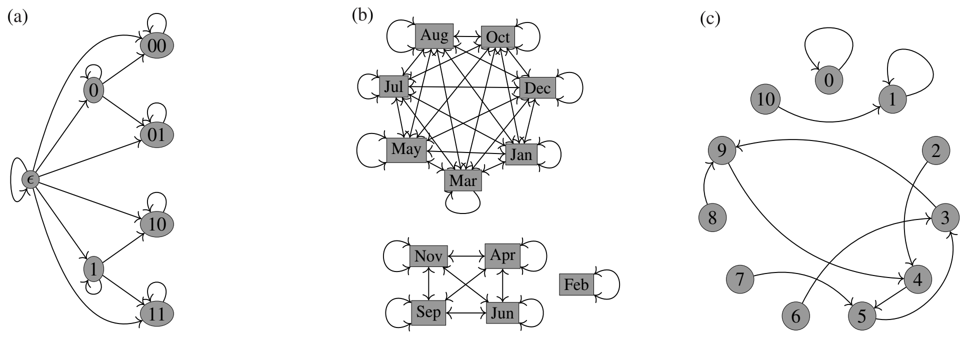


Figure 8.10 Visualizations of three relations: (a) *isPrefix* from Example 8.5 (prefixes of bitstrings); (b) months of the same length, from Example 8.11; and (c) $\langle x, x^2 \bmod 11 \rangle$ for $x \in \mathbb{Z}_{11}$, from Example 8.12.

consistent way (as in Figure 8.10a)—and we'll see these special types of relations in Section 8.4. But first we'll examine these three categories of properties in turn, and then we'll define *closures* of relations, which expand any relation R as little as possible while ensuring that the expansion of R has any particular desired subset of these properties.

8.3.1 Reflexivity

The *reflexivity* of a relation $R \subseteq A \times A$ is based on whether elements of A are related to themselves. (Latin: *re* "back" + *flect* "bend.") That is, are there pairs $\langle a, a \rangle$ in R? The relation R is *reflexive* if all of those $\langle a, a \rangle$ pairs are in R, and it's *irreflexive* if none of them are:

Definition 8.6: Reflexive and irreflexive relations.
A relation R on A is *reflexive* if, for every $x \in A$, we have that $\langle x, x \rangle \in R$. A relation R on A is *irreflexive* if, for every $x \in A$, we have that $\langle x, x \rangle \notin R$.

Using the visualization style from Figure 8.10, a relation is reflexive if every element $a \in A$ has a "loop" from a back to itself—and it's irreflexive if no $a \in A$ has a loop to itself. (See Figure 8.11.)

Example 8.17: Reflexivity of $=$, $\equiv_{17}$, *and* $\langle x, x^2 \rangle \bmod 11$.
Consider relations $=$ and $\equiv_{17}$ on $\mathbb{Z}$: that is, $\{\langle x, y \rangle : x = y\}$ and $\{\langle x, y \rangle : x \bmod 17 = y \bmod 17\}$. Both of these relations are reflexive, because $x = x$ and $x \bmod 17 = x \bmod 17$ for any $x \in \mathbb{Z}$.

But the relation $R = \left\{\langle x, x^2 \bmod 11 \rangle : x \in \mathbb{Z}_{11}\right\}$ from Figure 8.10c is not reflexive, because (among other examples) we have $\langle 7, 7 \rangle \notin R$.

Note that relations can be neither reflexive *nor* irreflexive. For example, the relation $S = \{\langle 0, 1 \rangle, \langle 1, 1 \rangle\}$ on $\{0, 1\}$ isn't reflexive (because $\langle 0, 0 \rangle \notin S$), but it's also not irreflexive (because $\langle 1, 1 \rangle \in S$).

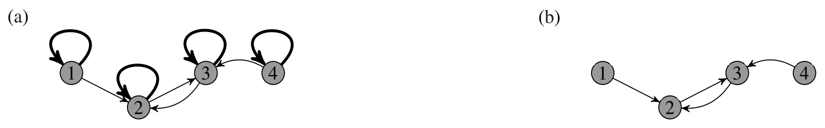


Figure 8.11 Reflexive and irreflexive relations: (a) a relation on A is reflexive if every $a \in A$ has a self-loop (the thick arrows); (b) a relation is irreflexive if no $a \in A$ has a self-loop.

Example 8.18: A few arithmetic relations.

Which of the following relations on $\mathbb{Z}^{\geq 1}$ are reflexive? Which are irreflexive?

$R_1 = \{\langle n,m\rangle : m \bmod n = 0\}$	divides
$R_2 = \{\langle n,m\rangle : n > m\}$	greater than
$R_3 = \{\langle n,m\rangle : n \leq m\}$	less than or equal to
$R_4 = \left\{\langle n,m\rangle : n^2 = m\right\}$	square
$R_5 = \{\langle n,m\rangle : n \bmod 5 = m \bmod 5\}$	equivalent mod 5

Solution. $|$ *is reflexive.* For any positive integer n, we have that $n \bmod n = 0$. Thus $\langle n,n\rangle \in R_1$ for any n.

$>$ *is irreflexive.* For any $n \in \mathbb{Z}^{\geq 1}$, we have that $n \not> n$. Thus $\langle n,n\rangle \notin R_2$ for any n.

$\leq$ *is reflexive.* For any positive integer n, we have $n \leq n$, so every $\langle n,n\rangle \in R_3$.

square *is neither reflexive nor irreflexive.* The square relation is not reflexive because $\langle 9,9\rangle \notin R_4$ and it is also not irreflexive because $\langle 1,1\rangle \in R_4$, for example. (That's because $9 \neq 9^2$, but $1 = 1^2$.)

$\equiv_5$ *is reflexive.* For any $n \in \mathbb{Z}^{\geq 1}$, we have $n \bmod 5 = n \bmod 5$, so $\langle n,n\rangle \in R_5$.

Note again that, as with *square*, it is possible to be *neither* reflexive *nor* irreflexive. (But it's not possible to be *both* reflexive *and* irreflexive, as long as $A \neq \varnothing$: for any specific $a \in A$, if $\langle a,a\rangle \in R$, then R is not irreflexive; if $\langle a,a\rangle \notin R$, then R is not reflexive.)

8.3.2 Symmetry

The *symmetry* of a relation $R \subseteq A \times A$ is based on whether the order of the elements in a pair matters. (Greek: *syn* "same" + *metron* "measure." Is R the same no matter which way you measure it?) That is, if the pair $\langle a,b\rangle$ is in R, is the pair $\langle b,a\rangle$ always also in R? (Or is it never in R? Or sometimes but not always?)

The relation R is *symmetric* if, for every a and b, the pairs $\langle a,b\rangle$ and $\langle b,a\rangle$ are both in R or both not in R. There are two accompanying notions: a relation R is *antisymmetric* if the only time $\langle a,b\rangle$ and $\langle b,a\rangle$ are both in R is when $a = b$, and R is *asymmetric* if $\langle a,b\rangle$ and $\langle b,a\rangle$ are never both in R (whether $a = b$ or $a \neq b$). Here are the formal definitions:

Definition 8.7: Symmetric, antisymmetric, and asymmetric relations.

- A relation R on A is *symmetric* if, for every $a \in A$ and $b \in A$, if $\langle a,b\rangle \in R$ then $\langle b,a\rangle \in R$.
- A relation R on A is *antisymmetric* if, for every $a \in A$ and $b \in A$ such that $\langle a,b\rangle \in R$ and $\langle b,a\rangle \in R$, we have $a = b$.
- A relation R on A is *asymmetric* if, for every $a \in A$ and $b \in A$, if $\langle a,b\rangle \in R$ then $\langle b,a\rangle \notin R$.

An important etymological note: *anti-* means "against" rather than "not." *Asymmetric* (there is no $\langle a,b\rangle \in R$ when $\langle b,a\rangle \in R$) is different from *antisymmetric* (whenever $\langle a,b\rangle \in R$ and $\langle b,a\rangle \in R$ then $a = b$) is different from *not symmetric* (there is some $\langle a,b\rangle \in R$ but $\langle b,a\rangle \notin R$).

Again thinking about our visualization of relations: a relation is symmetric if every arrow $a \to b$ is matched by an arrow $b \to a$ in the opposite direction. It's antisymmetric if there are no matched bidirectional pairs of arrows between two distinct elements a and b; and it's asymmetric if there also aren't even any self-loops. (An a-to-a self-loop is, in a weird way, a "pair" of arrows $a \to b$ and $b \to a$, just with $a = b$.) See Figure 8.12.

Example 8.19: Some symmetric relations.
The relations

$$\{\langle w, w'\rangle : w \text{ and } w' \text{ have the same length}\} \qquad \textit{(on the set of English words)}$$
$$\{\langle s, s'\rangle : s \text{ and } s' \text{ sat next to each other in class today}\} \qquad \textit{(on the set of students)}$$

are both symmetric. If w contains the same number of letters as w', then w' also contains the same number of letters as w. And if I sat next to you, then you sat next to me!

(The first relation is also reflexive—`ZEUGMA` contains the same number of letters as `ZEUGMA`—but the latter is irreflexive, as no student sits beside herself in class. [*zeugma*, n.: grammatical device in which words are used in parallel construction syntactically, but not semantically, as in *Yesterday, Alice caught a rainbow trout and hell from Bob for fishing all day.*])

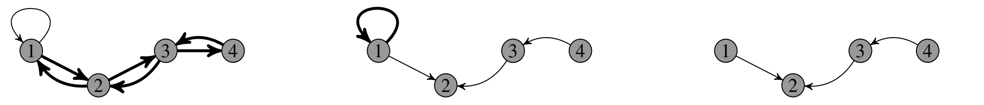
Example 8.20: A few arithmetic relations, again.
Which of these relations (from Example 8.18) are symmetric? Antisymmetric? Asymmetric?

$$\begin{aligned}
R_1 &= \{\langle n,m\rangle : m \bmod n = 0\} && \text{divides}\\
R_2 &= \{\langle n,m\rangle : n > m\} && \text{greater than}\\
R_3 &= \{\langle n,m\rangle : n \le m\} && \text{less than or equal to}\\
R_4 &= \left\{\langle n,m\rangle : n^2 = m\right\} && \text{square}\\
R_5 &= \{\langle n,m\rangle : n \bmod 5 = m \bmod 5\} && \text{equivalent mod 5}
\end{aligned}$$

Solution. $|$ *is antisymmetric.* Because $n \bmod m = 0$ and $m \bmod n = 0$ if and only if $n = m$, we know that if $\langle n,m\rangle \in R_1$ and $\langle m,n\rangle \in R_1$ then $n = m$. But the relation is neither symmetric (for example, $3 \mid 6$ but $6 \nmid 3$) nor asymmetric (for example, $3 \mid 3$).

$>$ *is asymmetric (and therefore antisymmetric).* If $x < y$ then $y \not< x$, even if $x = y$. So R_2 is asymmetric, which means that it is also antisymmetric.

$\le$ *is antisymmetric.* Similar to (1), R_3 is antisymmetric: if $x \le y$ and $y \le x$, then $x = y$. (But $3 \le 6$ and $6 \not\le 3$, and $3 \le 3$, so R_3 is neither symmetric nor asymmetric.)



(a) R is symmetric if every $a \to b$ is matched by $b \to a$.

(b) R is antisymmetric if no $a \leftrightarrow b$ exists for $a \neq b$. Self-loops (like the thick arrow) are allowed.

(c) R is asymmetric if it is antisymmetric and it also has no self-loops.

Figure 8.12 Symmetric, antisymmetric, and asymmetric relations.

square *is antisymmetric.* R_4 is not symmetric because $\langle 3, 9\rangle \in R_4$ but $\langle 9, 3\rangle \notin R_4$, and it's not asymmetric because $\langle 1, 1\rangle \in R_4$. (That's because $3^2 = 9$ but $9^2 \neq 3$, and $1^2 = 1$.) But it is antisymmetric: the only way $x^2 = y$ and $y^2 = x$ is if $x = y$ (specifically $x = y = 0$ or $x = y = 1$).

$\equiv_5$ *is symmetric.* The "equivalent mod 5" relation is symmetric because equality is: for any n and m, we have $n \bmod 5 = m \bmod 5$ if and only if $m \bmod 5 = n \bmod 5$. But it's not antisymmetric: $\langle 17, 202\rangle \in R_5$ and $\langle 202, 17\rangle \in R_5$.

Note that it is possible for a relation to be *both* symmetric and antisymmetric; see Exercise 8.70. And it's also possible for a relation R not to be symmetric, but also for R to fail to be either antisymmetric or asymmetric:

Example 8.21: A non-symmetric, non-asymmetric, non-antisymmetric relation.
The relation $R = \{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 0\rangle\}$ on $\{0, 1, 2\}$ isn't symmetric ($0 \to 2$ but $2 \not\to 0$), and it isn't asymmetric or antisymmetric ($0 \to 1$ and $1 \to 0$ but $0 \neq 1$).

One other useful way to think about the symmetry (or antisymmetry/asymmetry) of a relation R is by considering the inverse R^{-1} of R. Recall that R^{-1} reverses the direction of all of the arrows of R, so $\langle a, b\rangle \in R$ if and only if $\langle b, a\rangle \in R^{-1}$. A symmetric relation is one in which every $a \to b$ arrow is matched by a $b \to a$ arrow, so reversing the arrows doesn't change the relation. For an antisymmetric relation R, the inverse R^{-1} has only self-loops in common with R. And an asymmetric relation has no arrows in common with its inverse. (See Figure 8.13.) Specifically (see Exercises 8.67–8.69):

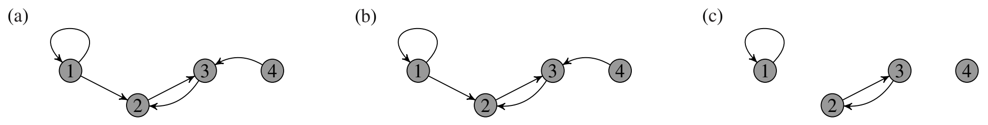


Figure 8.13 (a) A relation R, (b) its inverse R^{-1}, and (c) their intersection $R \cap R^{-1}$. From (c), we see that R isn't symmetric ($1 \to 2$ and $4 \to 3$ are missing), asymmetric (1 has a self-loop) or antisymmetric ($2 \leftrightarrow 3$ is present).

Theorem 8.8: Symmetry in terms of inverses.
Let $R \subseteq A \times A$ be a relation and let R^{-1} be its inverse. Then:

- R is symmetric if and only if $R \cap R^{-1} = R = R^{-1}$.
- R is antisymmetric if and only if $R \cap R^{-1} \subseteq \{\langle a, a \rangle : a \in A\}$.
- R is asymmetric if and only if $R \cap R^{-1} = \varnothing$.

8.3.3 Transitivity

The *transitivity* of a relation $R \subseteq A \times A$ is based on whether the relation always contains a "short circuit" from a to c whenever two pairs $\langle a, b \rangle$ and $\langle b, c \rangle$ are in R. (Latin: *trans* "across/through.") An alternative view is that a transitive relation R is one in which "applying R twice" doesn't yield any new connections. For example, consider the relation "lives in the same town as": if a person x lives in the same town as a person y you live in same town as, then in fact x directly (without reference to the intermediary y) lives in the same town as you. Here is the formal definition:

Definition 8.9: Transitive relation.
A relation R on A is *transitive* if, for every $a, b, c \in A$, if $\langle a, b \rangle \in R$ and $\langle b, c \rangle \in R$, then $\langle a, c \rangle \in R$ too.

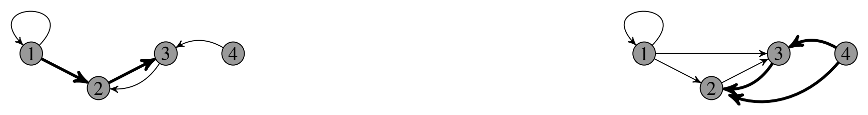
Or, using the visualization from Figure 8.10, a relation is transitive if there are no "open triangles": if $a \to b$ and $b \to c$, then $a \to c$. (In any "chain" of connected elements in a transitive relation, every element is also connected to all elements that are "downstream" of it.) See Figure 8.14.

Example 8.22: Some transitive relations.
The relations

$\{\langle w, w' \rangle : w \text{ and } w' \text{ have the same length}\}$ *(on the set of English words)*

$\{\langle s, s' \rangle : s \text{ arrived in class before } s' \text{ today}\}$ *(on the set of students)*

are both transitive. If w contains the same number of letters as w', and w' contains the same number of letters as w'', then w certainly contains the same number of letters as w'' too. And if Alice got to class before Bob, and Bob got to class before Charlie, then Alice got to class before Charlie.



(a) A relation that is not transitive (the thick arrows form an open triangle: $1 \to 3$ is missing).

(b) A transitive relation, with a highlighted closed triangle.

Figure 8.14 Transitivity of relations. A relation on A is transitive if every triangle is closed.

Example 8.23: A few arithmetic relations, one more time.
Which of the relations from Examples 8.18 and 8.20 are transitive?

$R_1 = \{\langle n, m\rangle : m \bmod n = 0\}$	divides
$R_2 = \{\langle n, m\rangle : n > m\}$	greater than
$R_3 = \{\langle n, m\rangle : n \le m\}$	less than or equal to
$R_4 = \left\{\langle n, m\rangle : n^2 = m\right\}$	square
$R_5 = \{\langle n, m\rangle : n \bmod 5 = m \bmod 5\}$	equivalent mod 5

Solution. $\mid$ *is transitive.* Suppose that $a \mid b$ and $b \mid c$. We need to show that $a \mid c$. But that's not too hard: by definition $a \mid b$ and $b \mid c$ mean that $b = ak$ and $c = b\ell$ for integers k and ℓ. Therefore $c = a \cdot (k\ell)$—and thus $a \mid c$. (This fact was Theorem 7.7.4.)

$>$ *is transitive.* If $x > y$ and $y > z$, then we know $x > z$.

$\le$ *is transitive.* Just as in (2), R_3 is transitive: if $x \le y$ and $y \le z$, then $x \le z$.

square *is not transitive.* The square relation isn't transitive, because, for example, we have $\langle 2, 4\rangle \in R_4$ and $\langle 4, 16\rangle \in R_4$—but $\langle 2, 16\rangle \notin R_4$. (That's because $2^2 = 4$ and $4^2 = 16$ but $2^2 \ne 16$.)

$\equiv_5$ *is transitive.* The "equivalent mod 5" relation is transitive because equality is: if $n \bmod 5 = m \bmod 5$ and $m \bmod 5 = p \bmod 5$, then $n \bmod 5 = p \bmod 5$.

While we can understand the transitivity of a relation R directly from Definition 8.9, we can also think about the transitivity of R by considering the relationship between R and $R \circ R$—that is, R and the composition of R with itself. (Earlier we saw how to view the symmetry of R by connecting R and its inverse R^{-1}.)

Theorem 8.10: Transitivity in terms of self-composition.
Let $R \subseteq A \times A$ be a relation. Then R is transitive if and only if $R \circ R \subseteq R$.

A proof of this theorem is deferred to the exercises (see Exercise 8.86).

Taking it further: Your preferences among a set of possibilities form a transitive relation: if you prefer chocolate ice cream to mint, and mint to strawberry, then you surely prefer chocolate to strawberry, too. But if you and a bunch of friends (all of whom may have different preferences about cuisines) want to go out to a restaurant, things become trickier; it's possible that your group's collective preferences may fail to be transitive (even if each individual's preferences are). The idea of a *voting system* is to determine the group's collective preferences based on its members' individual preferences, and there are some troubling paradoxes that arise in this voting context. (There's also the related issue of how to *implement* actual voting systems, perhaps electronically—while maintaining both secrecy and trust in the system.) See p. 8-38.

8.3.4 Properties of Asymptotic Relationships

Now that we've introduced the three categories of properties of relations (reflexivity, symmetry, and transitivity), let's consider one more set of relations in light of these properties: the *asymptotics* of functions. Recall from Chapter 6 that, for two functions $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ and $g : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$, we say that

$$\begin{array}{lll} f(n) \text{ is } O(g(n)) & \text{if and only if} & \exists n_0 \geq 0, c > 0 : [\forall n \geq n_0 : f(n) \leq c \cdot g(n)]. \\ f(n) \text{ is } \Theta(g(n)) & \text{if and only if} & f(n) \text{ is } O(g(n)) \text{ and } g(n) \text{ is } O(f(n)). \\ f(n) \text{ is } o(g(n)) & \text{if and only if} & f(n) \text{ is } O(g(n)) \text{ and } g(n) \text{ is not } O(f(n)). \end{array}$$

(Actually we previously phrased the definitions of $\Theta(\cdot)$ and $o(\cdot)$ in terms of $\Omega(\cdot)$, but the definition we've given here is completely equivalent, as proven in Exercise 6.30.) We can view these asymptotic properties as relations on the set $F = \left\{f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}\right\}$ of functions.

The standard asymptotic notation doesn't match the standard notation for relations—we write $f = \Theta(g)$ rather than $f \,\Theta\, g$ or $\langle f, g \rangle \in \Theta$—but Θ genuinely is a relation on F, in the sense that some pairs of functions are related by Θ and some pairs are not. And O and o are relations on F in the same way.

Example 8.24: O and Θ and o: reflexivity.

O is reflexive. For any function f, we can establish that $f = O(f)$ by choosing the constants $n_0 = 1$ and $c = 1$, because it is immediate that $\forall n \geq 1 : f(n) \leq 1 \cdot f(n)$. Therefore O is reflexive, because every function f satisfies $f = O(f)$.

Θ is reflexive. This fact follows immediately from the fact that O is reflexive:

$$\begin{array}{rll} \Theta \text{ is reflexive} & \Leftrightarrow \forall f \in F : f = \Theta(f) & \textit{definition of reflexivity} \\ & \Leftrightarrow \forall f \in F : f = O(f) \text{ and } f = O(f) & \textit{definition of } \Theta \\ & \Leftrightarrow \forall f \in F : f = O(f). & p \wedge p \equiv p \end{array}$$

But $\forall f \in F : f = O(f)$ is just the definition of O being reflexive, which we just established.

o is irreflexive. This fact follows by similar logic: for any function $f \in F$,

$$f = o(f) \Leftrightarrow f = O(f) \text{ and } f \neq O(f),$$

by the definition of $o(\cdot)$. But $p \wedge \neg p \equiv$ False (including when p is "$f = O(f)$"), so o is irreflexive.

Example 8.25: O and Θ and o: symmetry.

O is not symmetric, antisymmetric, or asymmetric. Define the functions $t_1(n) = n$ and $t_2(n) = n^2$ and $t_3(n) = 2n^2$. O is not symmetric because, for example, $t_1 = O(t_2)$ but $t_2 \neq O(t_1)$. O is not asymmetric because, for example, $t_1 = O(t_1)$. And O is not antisymmetric because, for example, $t_2 = O(t_3)$ and $t_3 = O(t_2)$ but $t_2 \neq t_3$.

Θ *is symmetric.* This fact follows immediately by definition: for arbitrary f and g,

$$\begin{aligned} f = \Theta(g) &\Leftrightarrow f = O(g) \text{ and } g = O(f) && \textit{definition of } \Theta \\ &\Leftrightarrow g = O(f) \text{ and } f = O(g) && p \wedge q \equiv q \wedge p \\ &\Leftrightarrow g = \Theta(f). && \textit{definition of } \Theta \end{aligned}$$

(Θ is not anti/asymmetric, because $t_2 = \Theta(t_3)$ for $t_2(n)$ and $t_3(n)$ defined above.)

o *is asymmetric.* This fact follows immediately, by similar logic: for arbitrary f and g, we have $f = o(g)$ and $g = o(f)$ if and only if $f = O(g)$ and $g \neq O(f)$ *and* $g = O(f)$ and $f \neq O(g)$—a contradiction! So if $f = o(g)$ then $g \neq o(f)$. Therefore o is asymmetric.

Exercises 6.18, 6.46, and 6.47 established that O, Θ, and o are all transitive.In sum: O is reflexive and transitive (but not symmetric, asymmetric, or antisymmetric); o is irreflexive, asymmetric, and transitive; and Θ is reflexive, symmetric, and transitive.

Taking it further: Among the computer scientists, philosophers, and mathematicians who study formal logic, there's a special kind of logic called *modal logic* that's of significant interest. Modal logic extends the type of logic we introduced in Chapter 3 to also include logical statements about whether a true proposition is *necessarily* true or *accidentally* true. For example, the proposition *Canada won the 2014 Olympic gold medal in curling* is true—but the gold-medal game *could* have turned out differently and, if it had, that proposition would have been false. But *Either it rained yesterday or it didn't rain yesterday* is true, and there's no possible scenario in which this proposition would have turned out to be false. We say that the former statement is "accidentally" true (it was an "accident" of fate that the game turned out the way it did), but the latter is "necessarily" true.

In modal logic, we evaluate the truth value of a particular logical statement multiple times, once in each of a set W of so-called *possible worlds*. Each possible world assigns truth values to every atomic proposition. Thus every logical proposition φ of the form we saw in Chapter 3 has a truth value in each possible world $w \in W$. But there's another layer to modal logic. In addition to the set W, we are also given a relation $R \subseteq W \times W$, where $\langle w, w' \rangle \in R$ indicates that w' *is possible relative to* w. In addition to the basic logical connectives from normal logic, we can also write two more types of propositions:

$\Diamond\varphi$	"possibly φ"	$\Diamond\varphi$ is true in w if $\exists w' \in W$ such that $\langle w, w' \rangle \in R$ and φ is true in w'.
$\Box\varphi$	"necessarily φ"	$\Box\varphi$ is true in w if $\forall w' \in W$ such that $\langle w, w' \rangle \in R$, φ is true in w'.

Of course, these operators can be nested, so we might have a proposition like $\Box(\Diamond p \Rightarrow \Box p)$.

Different assumptions about the relation R will allow us to use modal logic to model different types of interesting phenomena. For example, we might want to insist that $\Box\varphi \Rightarrow \varphi$ ("if φ is necessarily true, then φ is true": that is, if φ is true in every world $w' \in W$ possible relative to w, then φ is true in w). This axiom corresponds to the relation R being reflexive: w is always possible relative to w. Symmetry and transitivity correspond to the axioms $\varphi \Rightarrow \Box\Diamond\varphi$ and $\Box\varphi \Rightarrow \Box\Box\varphi$.

The general framework of modal logic (with different assumptions about R) has been used to represent logics of knowledge (where $\Box\varphi$ corresponds to "I know φ"); logics of provability (where $\Box\varphi$ corresponds to "we can prove φ"); and logics of possibility and necessity (where $\Box\varphi$ corresponds to "necessarily φ" and $\Diamond\varphi$ to "possibly φ"). Others have also studied *temporal logics* (where $\Box\varphi$ corresponds to "always φ" and $\Diamond\varphi$ to "eventually φ"); these logical formalisms have proven to be very useful in formally analyzing the correctness of programs. For a good introduction to modal logic, see [60].

8.3.5 Closures of Relations

Until now, in this section we've discussed some important properties that certain relations $R \subseteq A \times A$ may or may not happen to have. We'll close this section by looking at how to "force" the relation R to have one or more of these properties. Specifically, we will introduce the *closure* of a relation with respect to a property like symmetry: we'll take a relation R and expand it into a relation R' that has the desired property, while adding as few pairs to R as possible. That is, the *symmetric closure* of R is the smallest set $R' \supseteq R$ such that the relation R' is symmetric. Here are the formal definitions:

Definition 8.11: Reflexive, symmetric, and transitive closures.
Let $R \subseteq A \times A$ be a relation. Then:

The *reflexive closure of R* is the smallest relation $R' \supseteq R$ such that R' is reflexive.

The *symmetric closure of R* is the smallest relation $R'' \supseteq R$ such that R'' is symmetric.

The *transitive closure of R* is the smallest relation $R^+ \supseteq R$ such that R^+ is transitive.

Taking it further: In general, a set S is said to be *closed under the operation f* if, whenever we apply f to an arbitrary element of S (or to an arbitrary k-tuple of elements from S, if f takes k arguments), then the result is also an element of S. For example, the integers are closed under $+$ and $\cdot$, because the sum of two integers is always an integer, as is their product. But the integers are *not* closed under $/$: for example, $2/3$ is not an integer even though $2 \in \mathbb{Z}$ and $3 \in \mathbb{Z}$. The *closure* of S under f is the smallest superset of S that is closed under f.

We'll illustrate these definitions with a small example of symmetric, reflexive, and transitive closures, and then return to our running examples of arithmetic relations.

Example 8.26: Closures of a small relation.
Consider the relation $R = \{\langle 1,5\rangle, \langle 2,2\rangle, \langle 2,4\rangle, \langle 4,1\rangle, \langle 4,2\rangle\}$ on $\{1,2,3,4,5\}$. Then we have the following closures of R. (See Figure 8.15 for visualizations.)

$$\text{reflexive closure} = R \cup \left\{ \langle 1,1\rangle, \quad \langle 3,3\rangle, \quad \langle 4,4\rangle, \quad \langle 5,5\rangle \right\}$$

$$\text{symmetric closure} = R \cup \left\{ \underbrace{\langle 5,1\rangle}_{\text{because of } \langle 1,5\rangle}, \quad \underbrace{\langle 1,4\rangle}_{\text{because of } \langle 4,1\rangle} \right\}$$

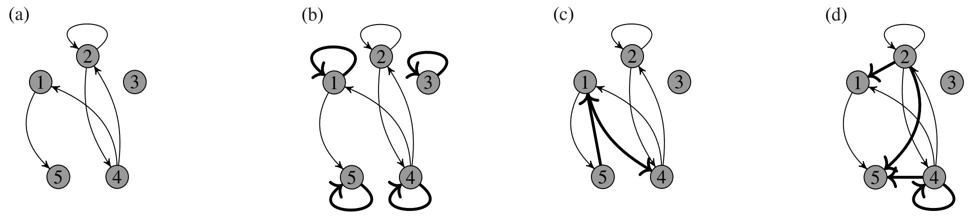


Figure 8.15 (a) A relation R, and several closures of R: (b) the reflexive closure; (c) the symmetric closure; and (d) the transitive closure. In each, the thick arrows had to be added to R to achieve the desired property.

$$\text{transitive closure} = R \cup \left\{ \underset{\substack{\text{because of}\\ \langle 2,4\rangle \text{ and } \langle 4,1\rangle}}{\langle 2,1\rangle,} \quad \underset{\substack{\text{because of}\\ \langle 4,2\rangle \text{ and } \langle 2,4\rangle}}{\langle 4,4\rangle,} \quad \underset{\substack{\text{because of}\\ \langle 4,1\rangle \text{ and } \langle 1,5\rangle}}{\langle 4,5\rangle,} \quad \underset{\substack{\text{because of}\\ \langle 2,4\rangle \text{ and } \langle 4,5\rangle}}{\langle 2,5\rangle} \right\}$$

It's worth noting that $\langle 2,5\rangle$ had to be in the transitive closure R^+ of R, even though there was no x such that $\langle 2,x\rangle \in R$ and $\langle x,5\rangle \in R$. There's one more intermediate step in the chain of reasoning: the pair $\langle 4,5\rangle$ had to be in R^+ because $\langle 4,1\rangle, \langle 1,5\rangle \in R$, and therefore both $\langle 2,4\rangle$ and $\langle 4,5\rangle$ had to be in R^+—so $\langle 2,5\rangle$ had to be in R^+ as well.

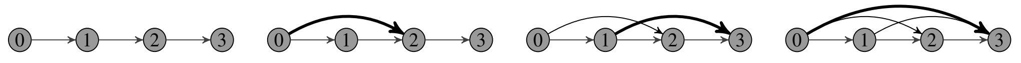
Example 8.27: Closures of divides.
Recall the "divides" relation $R = \{\langle n,m\rangle : m \bmod n = 0\}$. Because R is both reflexive and transitive, the reflexive closure and transitive closure of R are both just R itself. The symmetric closure of R is the set of pairs $\langle n,m\rangle$ where one of n and m is a divisor of the other: $\{\langle n,m\rangle : n \bmod m = 0 \text{ or } m \bmod n = 0\}$.

Example 8.28: Closures of $>$.
Recall the "greater than" relation $\{\langle n,m\rangle : n > m\}$. The reflexive closure of $>$ is $\geq$—that is, the set $\{\langle n,m\rangle : n \geq m\}$. The symmetric closure of $>$ is $\neq$—that is, the set $\{\langle n,m\rangle : n > m \text{ or } m > n\}$ is exactly $\{\langle n,m\rangle : n \neq m\}$. The relation $>$ is already transitive, so the transitive closure of $>$ is $>$ itself.

Computing the closures of a relation

How did we compute the closures in the last few examples? The approach itself isn't too hard: starting with $R' = R$, we repeatedly look for a violation of the desired property in R' (an element of R' required by the property but missing from R'), and repair that violation by adding the necessary element to R'. For the reflexive and symmetric closures, this idea is more straightforward: the violations of reflexivity are precisely those elements of $\{\langle a,a\rangle : a \in A\}$ not already in R, and the violations of symmetry are precisely those elements of R^{-1} that are not already in R.

For the transitive closure, things are slightly trickier: as we resolve existing violations by adding missing pairs to the relation, new violations of transitivity can crop up. (See Figure 8.16.) But to compute the transitive closure, we can simply iterate: starting with $R' := R$, repeatedly add to R' any missing $\langle a,c\rangle$ with $\langle a,b\rangle, \langle b,c\rangle \in R'$, until there are no more violations of transitivity. (While we won't prove it here, it's an important fact that the order in which we add elements to the transitive closure doesn't affect the final



(a) The relation R.

(b) $\langle 0,1\rangle$ and $\langle 1,2\rangle$ mean that we must add $\langle 0,2\rangle$.

(c) $\langle 1,2\rangle$ and $\langle 2,3\rangle$ mean that we must add $\langle 0,2\rangle$.

(d) $\langle 0,2\rangle$, which we added in (b), and $\langle 2,3\rangle$ mean that we must now add $\langle 0,3\rangle$ too.

We could have instead argued that we had to add $\langle 0,3\rangle$ because of $\langle 0,1\rangle$ and $\langle 1,3\rangle$ [from (c)] rather than because of $\langle 0,2\rangle$ [from (b)] and $\langle 2,3\rangle$.

Figure 8.16 Computing the transitive closure of the relation $\{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,3\rangle\}$.

```
reflexive-closure($R$):
Input: a relation $R \subseteq A \times A$
Output: the smallest reflexive $R' \supseteq R$
1 return $R \cup \{\langle a, a \rangle : a \in A\}$
```

```
symmetric-closure($R$):
Input: a relation $R \subseteq A \times A$
Output: the smallest symmetric $R' \supseteq R$
1 return $R \cup R^{-1}$
```

```
transitive-closure($R$):
Input: a relation $R \subseteq A \times A$
Output: the smallest transitive $R' \supseteq R$
1 $R' := R$
2 while there exist $a, b, c \in A$ such that
          $\langle a, b \rangle \in R$ and $\langle b, c \rangle \in R$ and $\langle a, c \rangle \notin R'$:
3     $R' := R' \cup \{\langle a, c \rangle\}$
4 return $R'$
```

Figure 8.17 Algorithms to compute the reflexive, symmetric, and transitive closures of $R \subseteq A \times A$, when A is finite.

result.) See Figure 8.17 for algorithms to compute these closures for $R \subseteq A \times A$ for a finite set A. (Note that these algorithms are *not* guaranteed to terminate if A is infinite! Also, there are faster ways to find the transitive closure based on graph algorithms—see Chapter 11—but the basic idea is captured here.)

Alternatively, here's another way to view the transitive closure of $R \subseteq A \times A$. The relation $R \circ R$ denotes precisely those pairs $\langle a, c \rangle$ where $\langle a, b \rangle, \langle b, c \rangle \in R$ for some $b \in A$. Thus the "direct" violations of transitivity are pairs that are in $R \circ R$ but not R. But, as we saw in Figure 8.16, the relation $R \cup (R \circ R)$ might have violations of transitivity, too: that is, a pair $\langle a, d \rangle \notin R \cup (R \circ R)$ but where $\langle a, b \rangle \in R$ and $\langle b, d \rangle \in R \circ R$ for some $b \in A$. So we have to add $R \circ R \circ R$ as well. And so on! In other words, the transitive closure R^+ of R is given by $R^+ = R \cup R^2 \cup R^3 \cup \cdots$, where $R^k = R \circ R \circ \cdots \circ R$ is the result of composing R with itself k times. Thus (see Exercise 8.105):

- the reflexive closure of R is $R \cup \{\langle a, a \rangle : a \in A\}$.
- the symmetric closure of R is $R \cup R^{-1}$.
- the transitive closure of R is $R \cup R^2 \cup R^3 \cup \cdots$.

Closures with respect to multiple properties at once

In addition to defining the closure of a relation R with respect to one of the three properties (reflexivity, symmetry, or transitivity), we can also define the closure with respect to two or more of these properties simultaneously. Any subset of these properties makes sense in this context, but the two most common combinations require reflexivity and transitivity, with or without requiring symmetry:

Definition 8.12: Reflexive [symmetric] transitive closure.
Let $R \subseteq A \times A$ be a relation. Then:

The *reflexive transitive closure of* R is the smallest relation $R^* \supseteq R$ such that R^* is both reflexive and transitive.

The *reflexive symmetric transitive closure of* R is the smallest relation $R^{\equiv} \supseteq R$ such that $R^{\equiv}$ is reflexive, symmetric, and transitive.

Example 8.29: Parent.
Consider the relation $parent = \{\langle p, c\rangle : p$ is a parent of $c\}$ over a set S. (This example makes sense if we think of S as a set of people where "parent" has biological meaning, or if we think of S as a set of nodes in a tree.) Then:

$$\begin{aligned}\text{transitive closure of } parent &= parent \cup grandparent \cup greatgrandparent \cup \cdots \\ \text{reflexive transitive closure of } parent &= yourself \cup parent \cup grandparent \cup greatgrandparent \cup \cdots\end{aligned}$$

where $yourself = \{\langle x, x\rangle : x \in S\}$. These closures of *parent* might be called *ancestor*: $\langle x, y\rangle$ is in the (reflexive) transitive closure of *parent* if and only if x is a direct ancestor of y. (The reflexive transitive closure counts you as an ancestor of yourself; the transitive closure does not.)

Example 8.30: Adjacent seating at a concert.
Consider a set S of people attending a concert held in a theater with rows of seats. Let R denote the relation of "sat immediately to the right of," so that $\langle x, y\rangle \in R$ if and only if x sat one seat to y's right in the same row. (See Figure 8.18.)

The transitive closure of R is "sat (not necessarily immediately) to the right of." The symmetric closure of R is "sat immediately next to." The symmetric transitive closure of R—just like the reflexive symmetric transitive closure—is "sat in the same row as." (You sit in the same row as yourself.)

As we discussed previously, we can think of the transitive closure R^+ of the relation R as the result of repeating R one or more times: in other words, $R^+ = R \cup R^2 \cup R^3 \cup \cdots$. The *reflexive* transitive closure of R also adds $\{\langle a, a\rangle : a \in A\}$ to the closure, which we can view as the result of repeating R *zero* or more times. In other words, we can write the reflexive transitive closure R^* as $R^* = R^0 \cup R^+$, where $R^0 = \{\langle a, a\rangle : a \in A\}$ represents the "zero-hop" application of R.

Taking it further: The basic idea underlying the (reflexive) transitive closure of a relation R—allowing (zero or) one or more repetitions of a relation R—also comes up in a widely useful tool for pattern matching in text, called *regular expressions*. Using regular expressions, you can search a text file for lines that match certain kinds of patterns (like: find all violations in the dictionary of the "I before E except after C" rule), or apply some operation to all files with a certain name (like: remove all `.txt` files). For more about regular expressions in general, and a little more on the connection between (reflexive) transitive closure and regular expressions, see p. 8-40.

We'll end with one last example of closures of an arithmetic relation:

Example 8.31: Closures of the successor relation.
The *successor* relation on the integers is $\{\langle n, n+1\rangle : n \in \mathbb{Z}\}$. What are the reflexive, symmetric, transitive, reflexive transitive, and reflexive symmetric transitive closures of this relation?

Solution. Perhaps the most interesting of these closures to think about is the transitive closure, which is best illustrated by the infinite version of Figure 8.16. For any n, we have $\langle n, n+1\rangle$ and $\langle n+1, n+2\rangle$ in

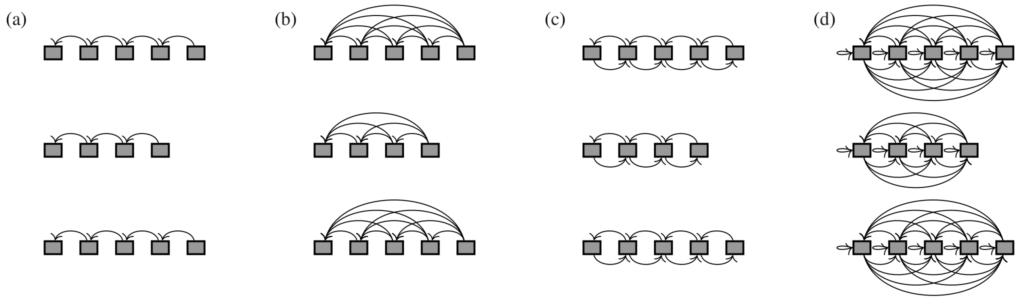


Figure 8.18 (a) The sat-immediately-to-the-right-of relation R for a three-row concert venue, and a few closures: (b) the transitive closure of R; (c) the symmetric closure of R; and (d) the symmetric transitive closure of R.

successor, so the transitive closure includes $\langle n, n+2 \rangle$. But $\langle n+2, n+3 \rangle$ is in *successor*, so the transitive closure also includes $\langle n, n+3 \rangle$. But $\langle n+3, n+4 \rangle$ is in *successor*, so the transitive closure also includes $\langle n, n+4 \rangle$. And so forth! (See Exercise 8.107 for a formal proof.) Here are the sets:

$$\begin{aligned}
\text{reflexive closure} &= \{\langle n, m \rangle : m = n \text{ or } m = n+1\} \\
\text{symmetric closure} &= \{\langle n, m \rangle : m = n-1 \text{ or } m = n+1\} \\
\text{transitive closure} &= \{\langle n, m \rangle : n < m\} \ \ \text{(that is, the relation } <) \\
\text{reflexive transitive closure} &= \{\langle n, m \rangle : n \leq m\} \ \ \text{(that is, the relation } \leq) \\
\text{reflexive symmetric transitive closure} &= \mathbb{Z} \times \mathbb{Z} \text{ (that is, \textit{every} pair of integers is in this relation).}
\end{aligned}$$

Taking it further: We can view $\leq$ (the reflexive transitive closure of *successor*) as either *the reflexive closure of* $<$ (the transitive closure of *successor*), or we can view $\leq$ as *the transitive closure of* $\{\langle n, m \rangle : m = n \text{ or } m = n+1\}$ (the reflexive closure of *successor*). (For any relation, the reflexive closure of the transitive closure equals the transitive closure of the reflexive closure.)

Computer Science Connections

What's Hard about Designing Voting Systems

Imagine a collection of n people who have individual preferences over k candidates. That is, we have n relations $R_1, R_2, \ldots, R_n$, each of which is a relation on the set $\{1, 2, \ldots, k\}$. We wish to aggregate these individual preferences into a single preference relation for the collection of people. Although this description is much more technical than our everyday usage, the problem that we've described here is well known: it's otherwise known as *voting.* (Economists also call this topic the theory of *social choice.*) But voting systems are hard to design, for at least two distinct reasons:

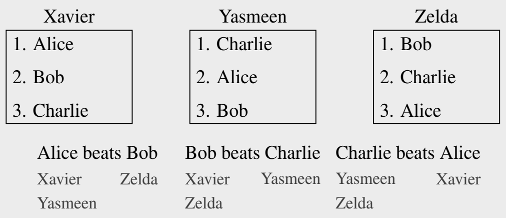


Figure 8.19 The Condorcet paradox: in head-to-head comparisons, Alice beats Bob, Bob beats Charlie, and Charlie beats Alice (each time a 2-to-1 victory).

Aggregating preferences. Some troubling paradoxes arise in voting problems, related to transitivity—or, more precisely, to the absence of transitivity. Figure 8.19 gives an example, for a three-candidate (Alice, Bob, Charlie) and three-voter (Xavier, Yasmeen, Zelda) election. (This paradox also arises when there are many more voters.) If the three voters have preferences as shown in the figure, then, in head-to-head runoffs between pairs of candidates, there's a "cycle" of winners. That's pretty weird: we have taken strict preferences (each of which is certainly transitive!) from each of the voters, and aggregated them into a nontransitive set of societal preferences. This phenomenon—no candidate would win a head-to-head vote against every other candidate—is called the *Condorcet paradox.* (The *Condorcet criterion* declares the winner of a vote to be the candidate who would win a runoff election against any other individual candidate. Here, no candidate satisfies the Condorcet criterion. Both of these notions are named after the French philosopher/mathematician Marquis de Condorcet [rhymes with *gone for hay*] (1743–1794).)

The Condorcet paradox is troubling, but an even more troubling result says that, more or less, there's *no* good way of designing a voting system! *Arrow's Theorem,* proven around 1950 by Kenneth Arrow (1921–2017)—an American economist who won the 1972 Nobel Prize in Economics, largely for this theorem [10]—states that there's no way to aggregate individual preferences to society-level preferences in a way that's consistent with three "obviously desirable" properties of a voting system: (1) if every voter prefers candidate A to candidate B, then A beats B; (2) there's no "dictator" (a single voter whose preferences of the candidates directly determines the outcome of the vote); and (3) "independence of irrelevant alternatives" (if candidate A beats B when candidate C is in the race, then A still beats B if C were to drop out of the race). The simplest voting system is "plurality" (or "relative majority") voting, in which every voter chooses their preferred candidate, and the winner is whichever candidate gets the most votes. But, increasingly, other voting schemes are used in some real-world elections, too: "ranked-choice" (or "instant-runoff") voting, where a voter ranks multiple candidates, or "Borda count" voting—named after Jean-Charles de Borda (1733–1799)—in which candidates receive different numbers of "points" from each voter. But what Arrow's Theorem says is that *all* of these voting systems sometimes exhibit some undesirable property.

Collecting and verifying the counting of votes. There has been increasing attention in recent years to the hardware, software, and procedures for the casting, processing, and tabulating of votes in elections. (In the U.S., this attention began to intensify after the "butterfly ballot" controversy in Florida in the 2000 presidential election, and further intensified during and after the 2020 presidential election. The design of the ballots themselves [did voters actually record their intended candidate on their ballot?] was the focus in 2000—the kinds of questions of human factors and usability that so often haunt software products, too; in 2020, the focus was more on the processes for acquiring, casting, and counting ballots.) It has been a long-standing temptation to try to develop an *electronic* voting system, perhaps using cryptographic protocols to enable individual voters to verify that their votes were cast and counted as they were intended. But the crucial fact that *ballots are meant to be secret* makes the cryptographic setting very different from traditional ones: if I can prove to you that my vote was recorded and counted for a particular candidate, you would be able to coerce or bribe me into voting in a particular way. Between this risk, and the inherent difficulty of designing computer systems that are truly secure, there is a clear, strong consensus among cryptography and security researchers advocating for (i) using paper ballots (and not electronic or, even worse, internet-based voting); and (ii) using statistical means (and not cryptographic means) to audit election results [94].

Computer Science Connections

Regular Expressions

Regular expressions (sometimes called *regexps* or *regexes* for short) are a mechanism to express pattern-matching searches in strings. (Their name is also a bit funny; more on that below.) Regular expressions are used by a number of useful utilities on Unix-based systems, like `grep` (which prints all lines of a file that match a given pattern) and `sed` (which can perform search-and-replace operations for particular patterns). And many programming languages have a capability for regular-expression processing; they're a tremendously handy tool for text processing.

Let Σ be an *alphabet* of symbols. (For convenience, think of $\Sigma = \{\texttt{A}, \texttt{B}, \ldots, \texttt{Z}\}$, but generally it's the set of all ASCII characters.) Let Σ^* denote the set of all finite-length strings of symbols from Σ. (The * notation echoes the notation for the reflexive transitive closure: Σ^* is the set of elements resulting from "repeating" Σ zero or more times.) Some of the basic syntax for regular expressions is shown in Figure 8.20, which recursively defines a relation $Matches \subseteq Regexps \times \Sigma^*$, where certain strings match a given pattern. For example, $\{s : \langle \texttt{A*B+}, s\rangle \in Matches\}$ is precisely the set of strings that can be written xy where $\langle \texttt{A*}, x\rangle$ and $\langle \texttt{B+}, y\rangle$ are in *Matches*. (And that means that strings like `AB` or `BBB` or `AAAAABBB`, with any number of `A`s and one or more `B`s, matches the regular expression `A*B+`.) A few other regexp operators correspond to the types of closures introduced in this section. There's some other shorthand for common constructions, too: for example, a list of characters in square brackets matches any of those characters (for example, `[AEIOU]` is shorthand for `(A|E|I|O|U)`). (Other syntax allows a range of characters or everything *but* a list of characters: for example, `[A-Z]` for all letters, and `[^AEIOU]` for consonants.)

regexp	*matched strings*
`A`	matches only the single character `A`
`B`	matches only the single character `B`
⋮	
`Z`	matches only the single character `Z`
`.`	any single character string
$\alpha\beta$	any string xy where x matches α and y matches β
α | β	any string x where x matches α *or* x matches β
α?	any string that matches α *or* the empty string
α+	any string $x_1x_2\ldots x_k$, with $k \geq 1$, where each x_i matches α
α*	matches any string $x_1x_2\ldots x_k$, with $k \geq 0$, where each x_i matches α

The + operator is roughly analogous to transitive closure: α+ matches any string that consists of one or more repetitions of α—while ? is roughly analogous to the reflexive closure and * to the reflexive transitive closure. The only difference is that here we're combining repetitions by *concatenation* rather than by *composition.*

Figure 8.20 Regular expressions: single characters, and regexp operators.

Figure 8.21 shows a few examples of regular expressions matching words in a dictionary with some vaguely interesting properties.

The odd-sounding name "regular expression" derives from a related notion, called a *regular language.* A *language* $L \subseteq \Sigma^*$ is a subset of all strings; in the subfield of theoretical computer science called *formal language theory,* we're interested in the difficulty of determining whether a

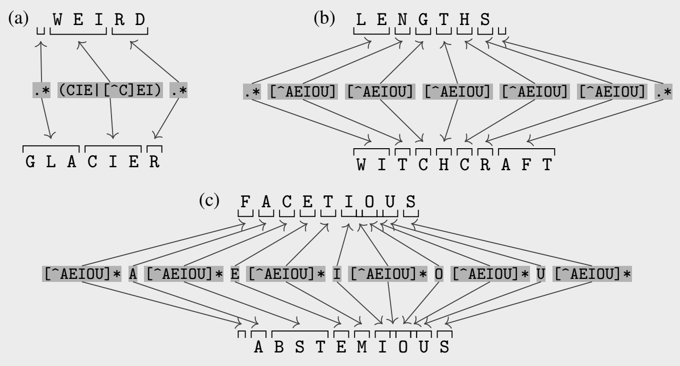

(a)
W E I R D
.* (CIE|[^C]EI) .*
G L A C I E R
(b)
L E N G T H S
.* [^AEIOU] [^AEIOU] [^AEIOU] [^AEIOU] [^AEIOU] .*
W I T C H C R A F T
(c)
F A C E T I O U S
[^AEIOU]* A [^AEIOU]* E [^AEIOU]* I [^AEIOU]* O [^AEIOU]* U [^AEIOU]*
A B S T E M I O U S

Figure 8.21 Three regular expressions, and two words that match each: (a) violations of "I before E except after C"; (b) words with five consecutive consonants; and (c) words with all five vowels, once each, in alphabetical order.

given string $x \in \Sigma^*$ is in L or not, for a particular language L. (Some example languages: the set of words containing only type of vowel, or the set of binary strings with the same number of `1`s and `0`s.) A *regular language* is one for which it's possible to determine whether $x \in L$ by reading the string from left to right and, at each step, remembering only a constant amount of information about what you've seen so far. (The set of univocalic words is regular; the set of "balanced" bitstrings is not.)

(For a bit more on regular languages, regular expressions, and formal language theory see p. 8-59; for a lot more, see a good textbook on computational complexity and formal languages, like [73] or [120].)

EXERCISES

8.51 Draw a directed graph representing the relation $\{\langle x, x^2 \bmod 13\rangle : x \in \mathbb{Z}_{13}\}$.

8.52 Repeat for $\{\langle x, 3x \bmod 13\rangle : x \in \mathbb{Z}_{15}\}$.

8.53 Repeat for $\{\langle x, 3x \bmod 15\rangle : x \in \mathbb{Z}_{15}\}$.

Which of the following relations on $\{0, 1, 2, 3, 4\}$ *are reflexive? Irreflexive? Neither?*

8.54 $\{\langle x, x\rangle : x^5 \equiv_5 x\}$

8.55 $\{\langle x, y\rangle : x + y \equiv_5 0\}$

8.56 $\{\langle x, y\rangle :$ there exists z such that $x \cdot z \equiv_5 y\}$

8.57 $\{\langle x, y\rangle :$ there exists z such that $x^2 \cdot z^2 \equiv_5 y\}$

Let $R \subseteq A \times A$ and $T \subseteq A \times A$ be relations. Prove or disprove:

8.58 Prove or disprove: R is reflexive if and only if R^{-1} is reflexive.

8.59 Prove or disprove: If R and T are both reflexive, then $R \circ T$ is reflexive.

8.60 Prove or disprove: If $R \circ T$ is reflexive, then R and T are both reflexive.

8.61 Prove or disprove: R is irreflexive if and only if R^{-1} is irreflexive.

8.62 Prove or disprove: If R and T are both irreflexive, then $R \circ T$ is irreflexive.

Which relations from Exercises 8.54–8.57 on $\{0, 1, 2, 3, 4\}$ *are symmetric? Antisymmetric? Asymmetric? Explain.*

8.63 $\{\langle x, x\rangle : x^5 \equiv_5 x\}$

8.64 $\{\langle x, y\rangle : x + y \equiv_5 0\}$

8.65 $\{\langle x, y\rangle :$ there exists z such that $x \cdot z \equiv_5 y\}$

8.66 $\{\langle x, y\rangle :$ there exists z such that $x^2 \cdot z^2 \equiv_5 y\}$

Prove Theorem 8.8, connecting the symmetry/asymmetry/antisymmetry of a relation R to the inverse R^{-1} of R.

8.67 Prove that R is symmetric if and only if $R \cap R^{-1} = R = R^{-1}$.

8.68 Prove that R is antisymmetric if and only if $R \cap R^{-1} \subseteq \{\langle a, a\rangle : a \in A\}$.

8.69 Prove that R is asymmetric if and only if $R \cap R^{-1} = \varnothing$.

8.70 Be careful: it's possible for a relation $R \subseteq A \times A$ to be both symmetric and antisymmetric! Describe, as precisely as possible, the set of relations on A that are both.

8.71 Use Theorem 8.8 to argue that every asymmetric relation is also antisymmetric.

Consider three possibilities for a relation's symmetry—symmetric, antisymmetric, and asymmetric—and also three possibilities for its reflexiveness: it can be reflexive, irreflexive, or neither. For each combination, identify a relation on $\{0, 1\}$ *that satisfies the given criteria, or, if the criteria are inconsistent, explain why there is no such relation.*

8.72 a reflexive, symmetric relation on $\{0, 1\}$

8.73 a reflexive, antisymmetric relation on $\{0, 1\}$

8.74 a reflexive, asymmetric relation on $\{0, 1\}$

8.75 an irreflexive, symmetric relation on $\{0, 1\}$

8.76 an irreflexive, antisymmetric relation on $\{0, 1\}$

8.77 an irreflexive, asymmetric relation on $\{0, 1\}$

8.78 a symmetric relation on $\{0, 1\}$ that's neither reflexive nor irreflexive

8.79 an antisymmetric relation on $\{0, 1\}$ that's neither reflexive nor irreflexive

8.80 an asymmetric relation on $\{0, 1\}$ that's neither reflexive nor irreflexive

Which relations from Exercises 8.54–8.57 on $\{0,1,2,3,4\}$ are transitive? Explain.

8.81 $\{\langle x,x\rangle : x^5 \equiv_5 x\}$.

8.82 $\{\langle x,y\rangle : x+y \equiv_5 0\}$.

8.83 $\{\langle x,y\rangle :$ there exists z such that $x \cdot z \equiv_5 y\}$.

8.84 $\{\langle x,y\rangle :$ there exists z such that $x^2 \cdot z^2 \equiv_5 y\}$.

8.85 Prove that, if R is irreflexive and transitive, then R is asymmetric.

8.86 Prove Theorem 8.10: show that R is transitive if and only if $R \circ R \subseteq R$.

8.87 Theorem 8.10 cannot be stated with an $=$ instead of $\subseteq$ (although I actually made this mistake in a previous draft of this chapter!). Give an example of a transitive relation R where $R \circ R \subset R$ (that is, where $R \circ R \neq R$).

The following exercises describe a relation with certain properties. For each, say whether it is possible for a relation $R \subseteq A \times A$ to simultaneously have all of the stated properties. If so, describe as precisely as possible what structure the relation R must have. If not, prove that it is impossible.

8.88 Is it possible for R to be simultaneously symmetric, transitive, and irreflexive?

8.89 Is it possible for R to be simultaneously transitive and a function?

8.90 Identify *all* relations R on $\{0,1\}$ that are transitive.

8.91 Of the transitive relations on $\{0,1\}$ from Exercise 8.90, which are also reflexive and symmetric?

Consider the relation $R = \{\langle 2,4\rangle, \langle 4,3\rangle, \langle 4,4\rangle\}$ on the set $\{1,2,3,4\}$.
Also consider the relation $T = \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 2,1\rangle, \langle 2,3\rangle, \langle 3,1\rangle, \langle 3,2\rangle, \langle 3,4\rangle, \langle 4,5\rangle\}$ on $\{1,2,3,4,5\}$.

8.92 What's the reflexive closure of R?

8.93 What's the symmetric closure of R?

8.94 What's the transitive closure of R?

8.95 What's the reflexive transitive closure of R?

8.96 What's the reflexive symmetric transitive closure of R?

8.97 What's the reflexive closure of T?

8.98 What's the symmetric closure of T?

8.99 What's the transitive closure of T?

8.100 What's the symmetric closure of $\geq$?

The next few exercises ask you to implement relations (and the standard relation operations) in a programming language of your choice. Don't worry too much about efficiency in your implementation; it's okay to run in time $\Theta(n^3)$, $\Theta(n^4)$ or even $\Theta(n^5)$ when relation R is on a set of size n.

8.101 *(programming required.)* Develop a basic implementation of relations on a set A. Also implement inverse (R^{-1}) and composition ($R \circ T$).

8.102 *(programming required.)* Write functions **reflexive?**, **irreflexive?**, **symmetric?**, **antisymmetric?**, **asymmetric?**, and **transitive?** to test whether a given relation R has the specified property.

8.103 *(programming required.)* Implement the closure algorithms (from Figure 8.17) for relations.

8.104 *(programming required.)* Using your implementation from Exercises 8.101–8.103, verify your answers to Exercises 8.72–8.80.

8.105 Prove that the transitive closure of R is indeed $R^+ = R \cup R^2 \cup R^3 \cup \cdots$, as follows: show that if $S \supseteq R$ is any transitive relation, then $R^k \subseteq S$. (We'd also need to prove that R^+ is transitive, but you can omit this part of the proof. You may find a recursive definition of R^k most helpful: $R^1 = R$ and $R^k = R \circ R^{k-1}$.)

8.106 Give an example of a relation $R \subseteq A \times A$, for a finite set A, such that the transitive closure of R contains at least $c \cdot |R|^2$ pairs, for some constant $c > 0$. Make c as big as you can.

8.107 Recall the relation *successor* $= \left\{\langle x, x+1\rangle : x \in \mathbb{Z}^{\geq 0}\right\}$. Prove by induction on k that, for any integer x and any positive integer k, we have that $\langle x, x+k\rangle$ is in the transitive closure of successor. (In other words, you're showing that the transitive closure of *successor* is $\geq$. You cannot rely on the algorithm in Figure 8.17 because $\mathbb{Z}^{\geq 0}$ is not finite!)

8.108 We talked about the X closure of a relation R, for X being any nonempty subset of the properties of reflexivity, symmetry, and transitivity. But we didn't define the "antisymmetric closure" of a relation R—with good reason! Why doesn't the antisymmetric closure make sense?

8.4 Special Relations: Equivalence Relations and Partial/Total Orders

At the return of consciousness, that closed
Before the pity of those two relations,
Which utterly with sadness had confused me,

New torments I behold, and new tormented
Around me, whichsoever way I move,
And whichsoever way I turn, and gaze.

Dante Alighieri (1265–1321)
The Divine Comedy, "Inferno: Canto VI" (1320)

In Section 8.3, we introduced three key categories of properties that a particular relation $R \subseteq A \times A$ might have: (ir)reflexivity, (a/anti)symmetry, and transitivity. Here we'll consider relations R that have one of two particular combinations of those three categories of properties. Two very different "flavors" of relations emerge from these two particular constellations of properties.

The first special type of relation is an *equivalence relation,* which is reflexive, symmetric, and transitive. Equivalence relations divide the elements of A into one or more groups of equivalent elements, so that all elements in the same group are "the same" under R.

The second special type of relation is an *order* relation, which is antisymmetric, transitive, and either reflexive or irreflexive. These relations "rank" the elements of A, so that some elements of A are "more R" than others.

In this section, we'll give formal definitions of these two types of relations, and look at a few applications.

8.4.1 Equivalence Relations

An *equivalence relation* $R \subseteq A \times A$ separates the elements of A into one or more groups, where any two elements in the same group are *equivalent* according to R:

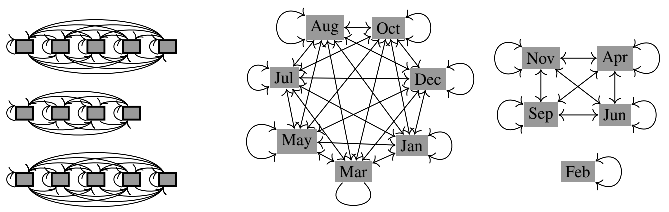
Definition 8.13: Equivalence relation.
An *equivalence relation* is a relation that is reflexive, symmetric, and transitive.

The most important equivalence relation that you've seen is equality ($=$): certainly, for any objects a, b, and c, we have that (i) $a = a$; (ii) $a = b$ if and only if $b = a$; and (iii) if $a = b$ and $b = c$, then $a = c$.

The relation *sat in the same row as* from Example 8.30 (see Figure 8.22a) is also an equivalence relation: it's reflexive (you sat in the same row as you yourself), symmetric (anyone you sat in the same row as also sat in the same row as you), and transitive (you sat in the same row as anyone who sat in the same row as someone who sat in the same row as you).

We've also previously seen another example in Example 8.11 (see Figure 8.22b for a reminder):

$$\{\langle m_1, m_2 \rangle : \text{months } m_1 \text{ and } m_2 \text{ have the same number of days (in some years)}\}$$



$$\left\{\begin{array}{l}\langle 0,0\rangle, \langle 0,12\rangle,\\ \langle 1,1\rangle, \langle 1,13\rangle,\\ \vdots\\ \langle 11,11\rangle, \langle 11,23\rangle\\ \langle 12,0\rangle, \langle 12,12\rangle\\ \langle 13,1\rangle, \langle 13,13\rangle\\ \vdots\\ \langle 23,11\rangle, \langle 23,23\rangle\end{array}\right\}$$

(a) "In the same row as," from Example 8.30.

(b) The months-of-the-same-length relation, from Example 8.11 and Figure 8.10b.

(c) Hours that are equivalent on a clock.

Figure 8.22 Reminders of two equivalence relations, and one new equivalence relation.

is also an equivalence relation. It's tedious but not hard to verify by checking all pairs that the relation is reflexive, symmetric, and transitive. (See also Exercises 8.116–8.118.)

Here are a few more examples of equivalence relations:

Example 8.32: Some equivalence relations.

The set of pairs from $\{0, 1, \ldots, 23\}$ with the same representation on a 12-hour clock is an equivalence relation. (See Figure 8.22c.)

The asymptotic relation Θ (that is, for two functions f and g, we have $\langle f, g\rangle \in \Theta$ if and only if f is $\Theta(g)$) is an equivalence relation. Example 8.24, Example 8.25, and Exercise 6.46 established that Θ is reflexive, symmetric, and transitive.

The relation $\equiv$ on logical propositions, where $P \equiv Q$ if and only if P and Q are true under precisely the same set of truth assignments, is an equivalence relation. (We even used the word "equivalent" in defining $\equiv$, which we called *logical equivalence* back in Chapter 3.)

Example 8.33: All equivalence relations on a small set.

List all equivalence relations on the set $\{\texttt{a}, \texttt{b}, \texttt{c}\}$.

Solution. There are five different equivalence relations on this set:

$\{\langle \texttt{a},\texttt{a}\rangle, \langle \texttt{b},\texttt{b}\rangle, \langle \texttt{c},\texttt{c}\rangle\}$	*"no element is equivalent to any other"*
$\{\langle \texttt{a},\texttt{a}\rangle, \langle \texttt{a},\texttt{b}\rangle, \langle \texttt{b},\texttt{a}\rangle, \langle \texttt{b},\texttt{b}\rangle, \langle \texttt{c},\texttt{c}\rangle\}$	*"*a *and* b *are equivalent, but they're different from* c*"*
$\{\langle \texttt{a},\texttt{a}\rangle, \langle \texttt{a},\texttt{c}\rangle, \langle \texttt{b},\texttt{b}\rangle, \langle \texttt{c},\texttt{a}\rangle, \langle \texttt{c},\texttt{c}\rangle\}$	*"*a *and* c *are equivalent, but they're different from* b*"*
$\{\langle \texttt{a},\texttt{a}\rangle, \langle \texttt{b},\texttt{b}\rangle, \langle \texttt{b},\texttt{c}\rangle, \langle \texttt{c},\texttt{b}\rangle, \langle \texttt{c},\texttt{c}\rangle\}$	*"*b *and* c *are equivalent, but they're different from* a*"*
$\{\langle \texttt{a},\texttt{a}\rangle, \langle \texttt{a},\texttt{b}\rangle, \langle \texttt{a},\texttt{c}\rangle, \langle \texttt{b},\texttt{a}\rangle, \langle \texttt{b},\texttt{b}\rangle, \langle \texttt{b},\texttt{c}\rangle, \langle \texttt{c},\texttt{a}\rangle, \langle \texttt{c},\texttt{b}\rangle, \langle \texttt{c},\texttt{c}\rangle\}$.	*"all elements are equivalent"*

Taking it further: A *deterministic finite automaton (DFA)* is a simple model of a so-called "machine" that has a finite amount of memory, and processes an input string by moving from state to state according to a fixed set of rules. DFAs can be used for a variety of applications (for example, in computer architecture, compilers, or in modeling simple behavior in computer games). And they can also be understood in terms of equivalence relations. See p. 8-59.

Equivalence classes

The descriptions of the quintet of equivalence relations on the set $\{\texttt{a}, \texttt{b}, \texttt{c}\}$ from Example 8.33 makes more explicit the other way that we've talked about an equivalence relation R on A: as a relation that carves up A into one or more *equivalence classes*, where any two elements of the same equivalence class are related by R (and no two elements of different classes are). Here's the formal definition:

Definition 8.14: Equivalence class.
Let $R \subseteq A \times A$ be an equivalence relation. The *equivalence class* of $a \in A$ is defined as the set $\{b \in A : \langle a, b \rangle \in R\}$ of elements related to A under R. The equivalence class of $a \in A$ under R is denoted by $[a]_R$—or, when R is clear from context, just as $[a]$.

The equivalence classes of an equivalence relation on A form a *partition* of the set A—that is, every element of A is in one and only one equivalence class. (See Definition 2.32 for the definition of a partition.)

Example 8.34: Equivalent mod 5.
Define the relation $\equiv_5$ on $\mathbb{Z}$, so that $\langle x, y \rangle \in \equiv_5$ if and only if $x \bmod 5 = y \bmod 5$. All three requirements of equivalence relations (reflexivity, symmetry, and transitivity) are met; see Examples 8.18, 8.20, and 8.23. There are five equivalence classes under $\equiv_5$:

$$\{0, 5, 10, \ldots\}, \{1, 6, 11, \ldots\}, \{2, 7, 12, \ldots\}, \{3, 8, 13, \ldots\}, \text{ and } \{4, 9, 14, \ldots\},$$

corresponding to the five possible values mod 5.

Example 8.35: Some equivalence classes.
The five different equivalence relations on $\{\texttt{a}, \texttt{b}, \texttt{c}\}$ in Example 8.33 correspond to five different sets of equivalence classes:

$\{\{\texttt{a}\}, \{\texttt{b}\}, \{\texttt{c}\}\}$	*"no element is equivalent to any other"*
$\{\{\texttt{a}, \texttt{b}\}, \{\texttt{c}\}\}$	*"*$\texttt{a}$ *and* $\texttt{b}$ *are equivalent, but they're different from* $\texttt{c}$*"*
$\{\{\texttt{a}, \texttt{c}\}, \{\texttt{b}\}\}$	*"*$\texttt{a}$ *and* $\texttt{c}$ *are equivalent, but they're different from* $\texttt{b}$*"*
$\{\{\texttt{a}\}, \{\texttt{b}, \texttt{c}\}\}$	*"*$\texttt{b}$ *and* $\texttt{c}$ *are equivalent, but they're different from* $\texttt{a}$*"*
$\{\{\texttt{a}, \texttt{b}, \texttt{c}\}\}.$	*"all elements are equivalent"*

An example: equivalence of rational numbers

Back in Chapter 2, we defined the rational numbers (that is, fractions) as the set $\mathbb{Q} = \mathbb{Z} \times \mathbb{Z}^{\neq 0}$: a rational number is a two-element sequence of integers (the numerator and the denominator, respectively), where the denominator is nonzero. (See Example 2.39.) But we haven't yet really talked about the fact that two rational numbers like $\langle 17, 34 \rangle$ and $\langle 101, 202 \rangle$ are equivalent, in the sense that $\frac{17}{34} = \frac{101}{202} = \frac{1}{2}$. Let's do that:

Example 8.36: Equivalence of rationals by reducing to lowest terms.
Formally define a relation $\equiv$ on $\mathbb{Q}$ that captures the notion of equality for fractions, and prove that $\equiv$ is an equivalence relation.

Solution. We define two rationals $\langle a, b\rangle$ and $\langle c, d\rangle$ as equivalent if and only if $ad = bc$—that is, we define the relation $\equiv$ as the set

$$\Big\{\big\langle\langle a, b\rangle, \langle c, d\rangle\big\rangle : ad = bc\Big\}.$$

To show that $\equiv$ is an equivalence relation, we must prove that $\equiv$ is reflexive, symmetric, and transitive. These three properties follow fairly straightforwardly from the fact that the relation $=$ on integers is an equivalence relation. We'll prove symmetry: for arbitrary $\langle a, b\rangle \in \mathbb{Q}$ and $\langle c, d\rangle \in \mathbb{Q}$, we have

$$\begin{aligned}
\langle a, b\rangle \equiv \langle c, d\rangle &\Rightarrow ad = bc && \textit{definition of} \equiv \\
&\Rightarrow bc = ad && \textit{symmetry of} = \\
&\Rightarrow \langle c, d\rangle \equiv \langle a, b\rangle. && \textit{definition of} \equiv
\end{aligned}$$

(Reflexivity and transitivity can be proven analogously.)

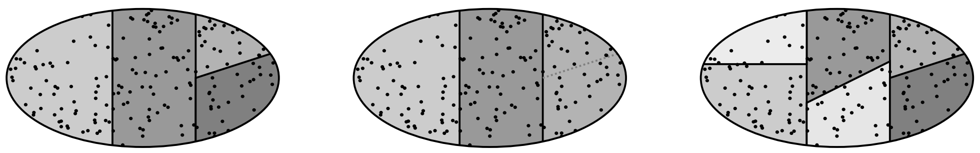
Taking it further: Recall that the equivalence class of a rational $\langle a, b\rangle \in \mathbb{Q}$ under $\equiv$, denoted $[\langle a, b\rangle]_\equiv$, represents the set of all rationals equivalent to $\langle a, b\rangle$. For example,

$$[\langle 17, 34\rangle]_\equiv = \{\langle 1, 2\rangle, \langle -1, -2\rangle, \langle 2, 4\rangle, \langle -2, -4\rangle, \ldots, \langle 17, 34\rangle, \ldots\}.$$

For equivalence relations like $\equiv$ for $\mathbb{Q}$, we may agree to associate an equivalence class with a *canonical element* of that class—here, the representative that's "in lowest terms." So we might agree to write $\langle 1, 2\rangle$ to denote the equivalence class $[\langle 1, 2\rangle]$, for example. This idea doesn't matter too much for the rationals, but it plays an important (albeit rather technical) role in figuring out how to define the real numbers in a mathematically coherent way. One standard way of defining the real numbers is as *the equivalence classes of converging infinite sequences of rational numbers,* called *Cauchy sequences* after the French mathematician Augustin Louis Cauchy (1789–1857). (Two converging infinite sequences of rational numbers are defined to be equivalent if they converge to the same limit—that is, if the two sequences eventually differ by less than ϵ, for all $\epsilon > 0$.) Thus when we write π, we're actually secretly denoting an infinitely large set of equivalent converging infinite sequences of rational numbers—but we're representing that equivalence class using a particular canonical form. Actually producing a coherent definition of the real numbers is a surprisingly recent development in mathematics, dating back less than 150 years. For more, see a good textbook on the subfield of math called *analysis.* (One classic book: [109].)

Coarsening and refining equivalence relations

An equivalence relation $\equiv$ on A slices up the elements of A into equivalence classes—that is, disjoint subsets of A such that any two elements of the same class are related by $\equiv$. For example, you might consider two restaurants equivalent if they serve food from the same cuisine (Thai, Indian, Ethiopian, Chinese, British, Minnesotan, . . .). But, given $\equiv$, we can imagine further subdividing the equivalence classes under $\equiv$ by making finer-grained distinctions (that is, *refining* $\equiv$)—perhaps dividing Indian into North Indian and South Indian, and Chinese into Americanized Chinese and Authentic Chinese. Or we could make $\equiv$ less specific



(a) An equivalence relation $\equiv$. Dots represent elements; each region denotes an equivalence class under $\equiv$.

(b) A coarsening of $\equiv$: a new equivalence relation formed by merging equivalence classes from $\equiv$.

(c) A refinement of $\equiv$: a new equivalence relation formed by subdividing $\equiv$'s equivalence classes.

Figure 8.23 Refining/coarsening an equivalence relation.

(that is, *coarsening* $\equiv$) by combining some equivalence classes—perhaps having only two equivalence classes, Delicious (Thai, Indian, Ethiopian, Chinese) and Okay (British, Minnesotan). See Figure 8.23.

Definition 8.15: Coarsening/refining equivalence relations.
Consider two equivalence relations $\equiv_c$ and $\equiv_r$ on the same set A. We say that $\equiv_r$ is a *refinement* of $\equiv_c$, or that $\equiv_c$ is a *coarsening* of $\equiv_r$, if $(a \equiv_r b) \Rightarrow (a \equiv_c b)$ for any $\langle a, b \rangle \in A \times A$. We can also refer to $\equiv_c$ as *coarser* than $\equiv_r$, and $\equiv_r$ as *finer* than $\equiv_c$.

For example, equivalence mod 10 is a refinement of equivalence mod 5: whenever $n \bmod 10 = m \bmod 10$ we know for certain that it will be the case that $n \bmod 5 = m \bmod 5$ too. (In other words, we have $(n \equiv_{10} m) \Rightarrow (n \equiv_5 m)$.) An equivalence class of the coarser relation is formed from the union of one or more equivalence classes of the finer relation. Here $\equiv_{10}$ is a refinement of $\equiv_5$, and, for example, the equivalence class $[3]_{\equiv_5}$ is the union of two equivalence classes from $\equiv_{10}$, namely $[3]_{\equiv_{10}} \cup [8]_{\equiv_{10}}$.

Example 8.37: Refining/coarsening equivalence relations on $\{\texttt{a}, \texttt{b}, \texttt{c}\}$.
In Example 8.35, we considered five different equivalence relations on $\{\texttt{a}, \texttt{b}, \texttt{c}\}$:

$\{\{\texttt{a}\}, \{\texttt{b}\}, \{\texttt{c}\}\}$	$\{\{\texttt{a}, \texttt{b}\}, \{\texttt{c}\}\}$	$\{\{\texttt{a}, \texttt{c}\}, \{\texttt{b}\}\}$	$\{\{\texttt{a}\}, \{\texttt{b}, \texttt{c}\}\}$	$\{\{\texttt{a}, \texttt{b}, \texttt{c}\}\}$
finest				coarsest

Of these, all three equivalence relations in the middle *refine* the one-class equivalence relation $\{\{\texttt{a}, \texttt{b}, \texttt{c}\}\}$ and *coarsen* the three-class equivalence relation $\{\{\texttt{a}\}, \{\texttt{b}\}, \{\texttt{c}\}\}$. (And the three-class equivalence relation $\{\{\texttt{a}\}, \{\texttt{b}\}, \{\texttt{c}\}\}$ also refines the one-class equivalence relation $\{\{\texttt{a}, \texttt{b}, \texttt{c}\}\}$.)

Taking it further: This is a very meta comment—sorry!—but we can think of "is a refinement of" as a relation *on equivalence relations on a set A*. In fact, the relation "is a refinement of" is reflexive, antisymmetric, and transitive: $\equiv$ refines $\equiv$; if $\equiv_1$ refines $\equiv_2$ and $\equiv_2$ refines $\equiv_1$ then $\equiv_1$ and $\equiv_2$ are precisely the same relation on A; and if $\equiv_1$ refines $\equiv_2$ and $\equiv_2$ refines $\equiv_3$ then $\equiv_1$ refines $\equiv_3$. Thus "is a refinement of" is, as per the definition to follow in the next section, a partial order on equivalence relations on the set A. That means, for example, that there is a *minimal element* according to the "is a refinement of" relation on the set of equivalence relations on any finite set A—that is, an equivalence relation $\equiv_{\min}$ such that $\equiv_{\min}$ is refined by no relation aside from $\equiv_{\min}$ itself. (Similarly, there's a maximal relation $\equiv_{\max}$ that refines no relation except itself.) See Exercises 8.119 and 8.120.

8.4.2 Partial and Total Orders

An equivalence relation $\equiv$ on a set A has properties that "feel like" a form of equality—differing from $=$ only in that there might be multiple elements that are unequal but nonetheless cannot be distinguished by $\equiv$. Here we'll introduce a different special type of relation, more akin to $\leq$ than $=$, that instead describes a consistent *order* among the elements of A:

Definition 8.16: Partial order.
Let A be a set. A relation $\preceq$ on A that is reflexive, antisymmetric, and transitive is called a *partial order*. (A relation $\prec$ on A that is *irreflexive,* antisymmetric, and transitive is called a *strict partial order*.)

(Actually, the requirement of antisymmetry in a strict partial order is redundant; see Exercise 8.85.) Here are a few examples, from arithmetic and sets:

Example 8.38: Some (strict) partial orders on $\mathbb{Z}$: $\mid$, $>$, and $\leq$.
In Examples 8.18, 8.20, and 8.23, we showed that the following relations are all antisymmetric, transitive, and either reflexive or irreflexive:

- divides (reflexive): $R_1 = \{\langle n, m\rangle : m \bmod n = 0\}$ is a partial order.
- greater than (irreflexive): $R_2 = \{\langle n, m\rangle : n > m\}$ is a strict partial order.
- less than or equal to (reflexive): $R_3 = \{\langle n, m\rangle : n \leq m\}$ is a partial order.

Example 8.39: The subset relation.
Consider the relation $\subseteq$ on the set $\mathscr{P}(\{0, 1\})$, which consists of the pairs of sets shown in Figure 8.24a. It's tedious but not hard to verify that $\subseteq$ is reflexive, antisymmetric, and transitive. (Perhaps the easiest way to see this fact is via Figure 8.24b, which abbreviates the visualizations in Figure 8.10 by leaving out an a-to-c arrow if their relationship is implied by transitivity because of a-to-b and b-to-c arrows. We'll see more of this type of abbreviated diagram in a moment.)

Comparability and total orders

In a partial order $\preceq$, there can be two elements $a, b \in A$ such that *neither $a \preceq b$ nor $b \preceq a$*: for the subset relation from Example 8.39 we have $\{0\} \not\subseteq \{1\}$ and $\{1\} \not\subseteq \{0\}$, and for the divides relation we have $17 \nmid 21$ and $21 \nmid 17$. In this case, the relation $\preceq$ does not say which of these elements is "smaller." This phenomenon is the reason that $\preceq$ is called a *partial* order, because it only specifies how *some* pairs compare.

Definition 8.17: Comparability.
Let $\preceq$ be a partial order on A. We say that two elements $a \in A$ and $b \in A$ are *comparable under* $\preceq$ if either $a \preceq b$ or $b \preceq a$. Otherwise we say that a and b are *incomparable.*

When there are no incomparable pairs under $\preceq$, then we call $\preceq$ a *total* order:

(a)
$$\begin{array}{ccc} \{\} \subseteq \{0\} & \{\} \subseteq \{1\} & \{\} \subseteq \{0,1\} \\ \{0\} \subseteq \{0\} & \{0\} \subseteq \{0,1\} & \\ \{1\} \subseteq \{1\} & \{1\} \subseteq \{0,1\} & \\ \{0,1\} \subseteq \{0,1\} & & \end{array}$$

(b)
$$\begin{tikzcd} & \{0,1\} & \\ \{0\} \arrow[ur] & & \{1\} \arrow[ul] \\ & \{\} \arrow[ul] \arrow[ur] & \end{tikzcd}$$

For any $A \in \mathscr{P}(0,1)$ and any $B \in \mathscr{P}(0,1)$, we have that $A \subseteq B$ if and only if we can get from A to B by following zero or more arrows in this diagram.

Figure 8.24 The $\subseteq$ relation on $\mathscr{P}(\{0,1\})$: (a) all 8 pairs of sets in the $\subseteq$ relation, and (b) a diagrammatic representation.

Definition 8.18: Total order.
A relation $\preceq$ on A is a *total order* if it's a partial order and every pair of elements in A is comparable. (A relation $\prec$ is a *strict total order* if $\prec$ is a strict partial order and every pair of *distinct* elements in A is comparable.)

Warning! It's easy to get muddled about incomparability because of common-language use of the word that's related to the technical definition but misleadingly different. In everyday usage, people may say "incomparable" (or "beyond compare") to mean "unequaled"—as in *Cheese from France is incomparable to cheese from Wisconsin.* Be careful! In the context of our work, "incomparable" means "cannot be compared" and *not* "cannot be matched."

A few examples of partial and total orders

Here are a few examples of orders, related to strings and to asymptotics:

Example 8.40: Ordering strings.
Which of the following relations (on the set of all [finite-length] strings of letters) are partial orders? Which are total orders? Of those that are partial or total orders, which are strict?

$\langle x, y \rangle \in R$ if $|x| \geq |y|$. (The length of a string x—the number of letters in x—is denoted $|x|$.)
$\langle x, y \rangle \in S$ if x comes alphabetically no later than y. (See Example 3.48.)
$\langle x, y \rangle \in T$ if the number of `A`s in x is smaller than the number of `A`s in y.

Solution. *String length.* The relation $\{\langle x, y \rangle : |x| \geq |y|\}$ is reflexive and transitive, but it is not antisymmetric: for example, both $\langle \texttt{PASCAL}, \texttt{RASCAL} \rangle$ and $\langle \texttt{RASCAL}, \texttt{PASCAL} \rangle$ are in the relation, but $\texttt{RASCAL} \neq \texttt{PASCAL}$. So this relation isn't a partial order.

Alphabetical order. The relation "comes alphabetically no later than" is reflexive (every word w comes alphabetically no later than w), antisymmetric (the only word that comes alphabetically no later than w *and* no earlier than w is w itself), and transitive (if w_1 is alphabetically no later than w_2 and w_2 is no later than w_3, then indeed w_1 is no later than w_3). Thus S is a partial order.

In fact, any two words are comparable under S: either w is a prefix of w' (and $\langle w, w' \rangle \in S$) or there's a smallest index i in which $w_i \neq w'_i$ (and either $\langle w, w' \rangle \in S$ or $\langle w', w \rangle \in S$, depending on whether w_i is earlier or later in the alphabet than w'_i). Thus S is actually a total order.

Number of As. The relation "contains fewer As than" is irreflexive (any word w contains exactly the same number of As as it contains, not *fewer* than that!) and transitive (if we have $a_w < a_{w'}$ and $a_{w'} < a_{w''}$, then we also have $a_w < a_{w''}$). Therefore the relation is antisymmetric (by Exercise 8.85), and thus T is a strict partial order. But neither $\langle \texttt{PASCAL}, \texttt{RASCAL} \rangle$ nor $\langle \texttt{RASCAL}, \texttt{PASCAL} \rangle$ are in T—both words contain two As, so neither has fewer than the other—and thus RASCAL and PASCAL are incomparable, and T is not a (strict) total order.

Example 8.41: O and o as orders.
We've argued that o is irreflexive (Example 8.24), transitive (Exercise 6.47), and asymmetric (Example 8.25). Thus o is a strict partial order. But o is *not* a (strict) total order: we saw a function $f(n)$ in Example 6.6 such that $f(n) \neq o(n^2)$ *and* $n^2 \neq o(f(n))$, so these two functions are incomparable.

And, though we showed that O is reflexive and transitive (Exercise 6.18), we showed that O is *not* antisymmetric (Example 8.25), because, for example, the functions $f(n) = n^2$ and $g(n) = 2n^2$ are O of each other. Thus O is not a partial order.

Taking it further: A relation like O that is both reflexive and transitive (but not necessarily antisymmetric) is sometimes called a *preorder*. Although O is not a partial order, it very much has an "ordering-like" feel to it: it *does* rank functions by their growth rate, but there are clusters of functions that are all equivalent under O. We can think of O as defining *a partial order on the equivalence classes under* Θ. We saw another preorder in Example 8.40, with the relation R ("x and y have the same length"): although there are many pairs of distinct strings x and y where $\langle x, y \rangle \in R$ and $\langle y, x \rangle \in R$, it is only because of ties in lengths that R fails to be a partial order—or, indeed, a total order.

Hasse diagrams

Let R be any relation on A. For $k \geq 1$, we will call a sequence $\langle a_1, a_2, \ldots, a_k \rangle \in A^k$ a *cycle* if $\langle a_1, a_2 \rangle, \langle a_2, a_3 \rangle, \cdots, \langle a_{k-1}, a_k \rangle \in R$ and $\langle a_k, a_1 \rangle \in R$. A cycle is a sequence of elements, each of which is related by R to the next element in the sequence (where the last element is related to the first). For a partial order $\preceq$, there are cycles with $k = 1$ (because a partial order is reflexive, $a_1 \preceq a_1$ for any a_1), but there are no longer cycles. (See Exercise 8.133.)

Recall the "directed graph" visualization of a relation $R \subseteq A \times A$ that we introduced earlier (see Figure 8.10): we write down every element of A, and then, for every pair $\langle a_1, a_2 \rangle \in R$, we draw an arrow from a_1 to a_2. For a relation R that's a partial order, we'll introduce a simplified visualization, called a *Hasse diagram*, that allows us to figure out the full relation R but makes the diagram dramatically cleaner. (Hasse diagrams are named after Helmut Hasse (1898–1979), a German mathematician.)

Let $\preceq$ be a partial order. Consider three elements a, b, and c such that $a \preceq b$ and $b \preceq c$ and $a \preceq c$. Then *the very fact that* $\preceq$ *is a partial order* means that $a \preceq c$ can be inferred from the fact that $a \preceq b$ and $b \preceq c$. (That's just transitivity.) Thus we will omit from the diagram any arrows that can be inferred via transitivity. Similarly, we will leave out self-loops, which can be inferred from reflexivity. Finally, as we discussed above, there are no nontrivial cycles (that is, there are no cycles other than self-loops) in a

(a)

$$\left\{\begin{array}{l}\langle 0,0\rangle, \langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 0,4\rangle, \\ \langle 1,1\rangle, \\ \langle 2,2\rangle, \langle 2,3\rangle, \langle 2,4\rangle, \\ \langle 3,3\rangle, \langle 3,4\rangle, \\ \langle 4,4\rangle \end{array}\right\}$$

(b)

$$\begin{tikzcd} & & 4 \arrow[d, dash] \\ & & 3 \arrow[d, dash] \\ 1 \arrow[dr, dash] & & 2 \arrow[dl, dash] \\ & 0 & \end{tikzcd}$$

Note that we've omitted all arrow directions (they all point up), all five self-loops (they can be inferred from reflexivity), and the pairs $\langle 0,3\rangle$, $\langle 0,4\rangle$, and $\langle 2,4\rangle$ (they can be inferred from transitivity).

Figure 8.25 (a) A partial order, and (b) a Hasse diagram representing it.

partial order. Thus we will arrange the elements so that when $a \preceq b$ we will draw *a physically below b* in the diagram; all arrows will implicitly point upward in the diagram.

A small partial order, and a Hasse diagram for it, are shown in Figure 8.25. Here is a somewhat larger example:

Example 8.42: Hasse diagram for divides.

A Hasse diagram for the relation $|$ (divides) on the set $\{1, 2, \ldots, 32\}$ is shown in Figure 8.26. Again, the diagram omits arrow directions, self-loops, and "indirect" connections that can be inferred by transitivity. For example, the fact that $2 \mid 20$ is implicitly represented by the arrows $2 \to 4 \to 20$ (or $2 \to 10 \to 20$).

Which arrows must be shown in a Hasse diagram? We must include all arrows that cannot be inferred by the definition of a partial order—in other words, we must draw a direct connections for all those relationships that are not "short circuits" of pairs of other relationships. In other words, we must draw lines for

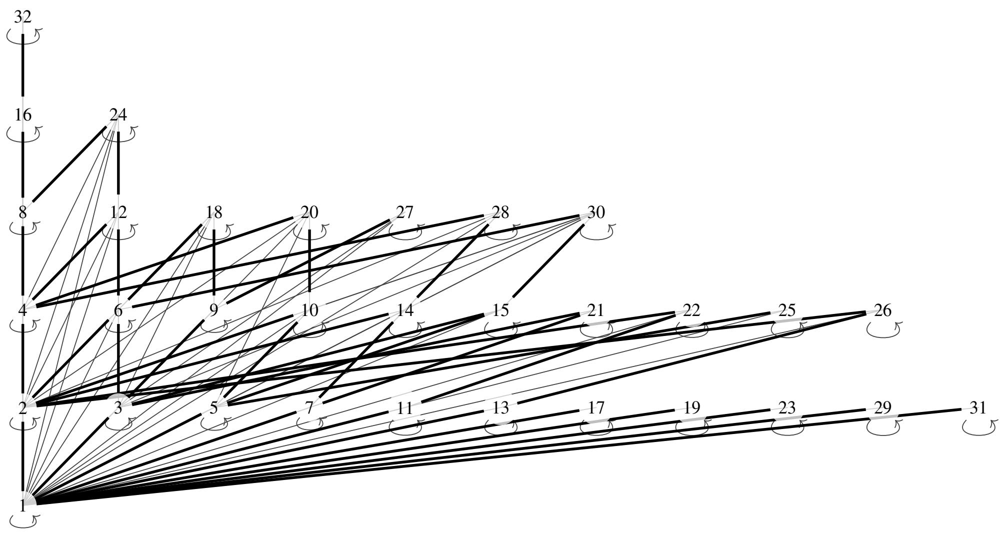


Figure 8.26 A Hasse diagram for "divides" on $\{1, 2, \ldots, 32\}$. The darker lines represent the Hasse diagram; the lighter arrows give the full picture of the relation, including all of the relationships that can be inferred from the fact that the relation is a partial order.

all those pairs $\langle a, c\rangle$ where $a \preceq c$ *and there is no* $b \notin \{a, c\}$ *such that* $a \preceq b$ *and* $b \preceq c$. Such a c is called an *immediate successor* of a.

Warning! When $a \preceq b$ holds for a partial order $\preceq$, we think of a as "smaller" than b under $\preceq$—a view that can be a little misleading if, for example, the partial order in question is $\geq$ instead of $\leq$. One example of this oddity: for $\geq$, the immediate successor of 42 is 41.

Minimal/maximal elements in a partial order

Consider the partial order $\preceq = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 2,2\rangle, \langle 2,4\rangle, \langle 3,3\rangle, \langle 4,4\rangle\}$—that is, the divides relation on the set $\{1, 2, 3, 4\}$. There's a strong sense in which 1 is the "smallest" element under $\preceq$: *every* element a satisfies $1 \preceq a$. And there's a slightly weaker sense in which 3 and 4 are both "largest" elements under $\preceq$: *no* element a satisfies $3 \preceq a$ or $4 \preceq a$. These ideas inspire two related pairs of definitions:

Definition 8.19: Minimum/maximum element.
For a partial order $\preceq$ on A:

- a *minimum element* is $x \in A$ such that, for every $y \in A$, we have $x \preceq y$.
- a *maximum element* is $x \in A$ such that, for every $y \in A$, we have $y \preceq x$.

Definition 8.20: Minimal/maximal element.
For a partial order $\preceq$ on A:

- a *minimal element* is $x \in A$ such that, for every $y \in A$ with $y \neq x$, we have $y \not\preceq x$.
- a *maximal element* is $x \in A$ such that, for every $y \in A$ with $y \neq x$, we have $x \not\preceq y$.

A maxim*al* whatzit is any whatzit that loses its whatzitness if we add anything to it. A maxim*um* whatzit is the largest possible whatzit. If you've studied calculus, you've seen a similar distinction under a different name: *maximal* corresponds to a local maximum; *maximum* corresponds to a global maximum.

Note that x being a minimal element does *not* demand that every other element be larger than x—only that no element is smaller! (Again, we're talking about a *partial* order—so $x \not\preceq y$ doesn't imply that $y \preceq x$.) In other words, a minimal element is one for which every other element y either satisfies $x \preceq y$ *or* is incomparable to x.

Example 8.43: Minimal/maximal/maximum/minimum elements in "divides".
For the divides relation on $\{1, 2, \ldots, 32\}$ (Example 8.42 and Figure 8.26):

- 1 is a minimum element. (Every $n \in \{1, 2, \ldots, 32\}$ satisfies $1 \mid n$.)
- 1 is also a minimal element. (No $n \in \{1, 2, \ldots, 32\}$ satisfies $n \mid 1$, except $n = 1$ itself.)
- There is no maximum element. (No $n \in \{1, 2, \ldots, 32\}$ aside from 32 satisfies $n \mid 32$, so 32 is the only candidate—but $31 \nmid 32$.)

- There are a slew of maximal elements: each of $\{17, 18, \ldots, 32\}$ is a maximal element. (None of these elements divides any $n \in \{1, 2, \ldots, 32\}$ other than itself.)

(You'll prove that any minimum element is also minimal, and that there can be at most one minimum element in a partial order, in Exercises 8.148 and 8.149.)

We've already seen some small partial orders that don't have minimum or maximum elements, but every partial order over a finite set must have at least one minimal element and at least one maximal element:

Theorem 8.21: Every (finite) partial order has a minimal/maximal element.
Let $\preceq \subseteq A \times A$ be a partial order on a finite set A. Then $\preceq$ has at least one minimal element and at least one maximal element.

Proof. We'll prove that there's a minimal element; the proof for the maximal element is analogous. Our proof is constructive; we'll give an algorithm to *find* a minimal element:

```
• let $i := 1$, and let $x_1$ be an arbitrarily chosen element in $A$.
• while there exists any $y \neq x_i$ with $y \preceq x_i$:
    let $x_{i+1}$ be any such $y$ (that is, one with $y \neq x_i$ and $y \preceq x_i$), and then let $i := i + 1$.
• return $x_i$.
```

It's not too hard to see that *if this algorithm terminates, then it returns a minimal element.* After all, the while loop only terminates when we've found an $x_i \in A$ such that there's no $y \neq x_i$ with $y \preceq x_i$—which is precisely the definition of x_i being a minimal element. Thus the real work is in proving that this algorithm actually terminates.

We claim that after $|A|$ iterations of the **while** loop—that is, after we've defined $x_1, x_2, \ldots, x_{|A|+1}$—we must have found a minimal element. Suppose not. Then we have found elements $x_1 \succeq x_2 \succeq \cdots \succeq x_{|A|+1}$, where $x_{i+1} \neq x_i$ for each i. Because there are only $|A|$ different elements in A, in a sequence of $|A| + 1$ elements we must have encountered the same element more than once. (This argument implicitly makes use of the *pigeonhole principle,* which we'll see in much greater detail in Chapter 9.) But that's a cycle containing two or more elements! And Exercise 8.133 asks you to show that there are no such cycles in a partial order. $\square$

Note that Theorem 8.21 only claimed that a minimal element must exist in a partial order *on a finite set* A. The claim would be false without that assumption! If A is an infinite set, then there may be no minimal element in A under a partial order. (See Exercise 8.146.)

We can identify minimal and maximal elements of a partial order directly from the Hasse diagram: they're simply the elements that aren't connected to anything above them (the maximal elements), and those that aren't connected to anything below them (the minimal elements). And, indeed, there are always topmost element(s) and bottommost element(s) in a Hasse diagram, and thus there are always maximal/minimal elements in any partial order—if the set of elements is finite, at least!

Problem-solving tip: A good visualization of data can make an apparently complicated statement much simpler. Another way to state Theorem 8.21 and its proof: start anywhere, and follow lines downward in the Hasse diagram; eventually, you must run out of elements below you, and you can't go any lower. Thus there's at least one bottommost element in any (finite) Hasse diagram.

8.4.3 Topological Ordering

Partial orders can be used to specify constraints on the order in which certain tasks must be completed: the printer must be loaded with paper before the document can be printed; the document must be written before the document can be printed; the paper must be purchased before the printer can be loaded with paper. Or, as another example: a computer science major at a certain college must take courses following the prerequisite structure specified in Figure 8.27.

But, while these types of constraints impose on a *partial* order on elements, the tasks must actually be completed in some sequence. (Likewise, the courses must be taken in some sequence—for a major who avoids "doubling up" on CS courses in the same term, at least.) The challenge we face is to *extend* a partial order into a total order—that is, to create a total order that obeys all of the constraints of the partial order, while making comparable all previously incomparable pairs.

Definition 8.22: Consistency of a total order with a partial order.
A total order $\preceq_{\text{total}}$ is *consistent with the partial order* $\preceq$ if $a \preceq b$ implies that $a \preceq_{\text{total}} b$.

In general, there are many total orders that are consistent with a given partial order. Here's an example:

Example 8.44: Ordering CS classes.
Here are two (of many!) course orderings that are consistent with the prerequisites in Figure 8.27:

Order A: (1) intro to CS, (2) data structures, (3) math of CS, (4) intro to computer systems, (5) software design, (6) programming languages, (7) algorithms, and (8) computability & complexity.

Order B: (1) intro to CS, (2) data structures, (3) software design, (4) programming languages, (5) math of CS, (6) algorithms, (7) computability & complexity, (8) intro to computer systems.

Order A corresponds to reading the elements of the Hasse diagram from the bottom-to-top (and left-to-right within a "row"); Order B corresponds to completing the top row left-to-right (first recursively completing the requirements to make the next element of the top row valid).

As in these examples, we can construct a total order that's consistent with any given partial order on the set A. Such an ordering of A is called a *topological ordering* of A. (Some people will refer to a topological ordering as a *topological sort* of A.) We'll prove this result inductively, by repeatedly identifying a minimal element a from the set of unprocessed elements, and then adding constraints to make a be a minim*um* element (and not just a minim*al* element).

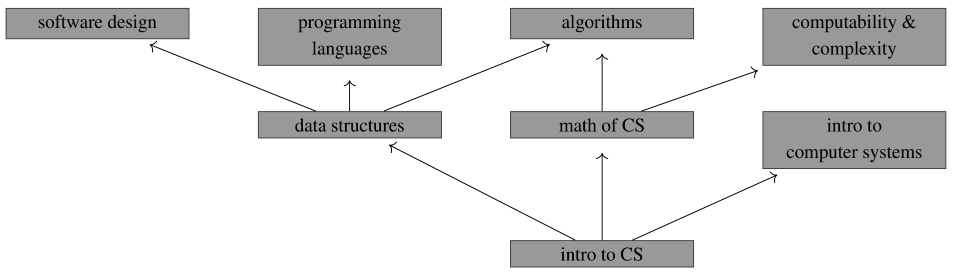


Figure 8.27 The CS major at a certain college in the midwest.

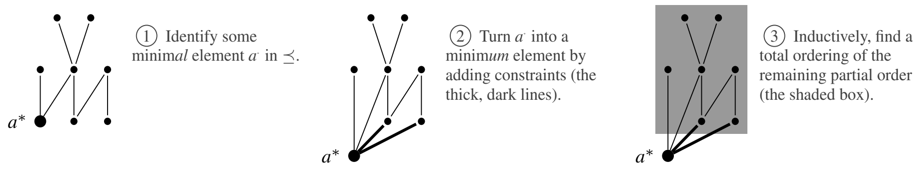

a^*
① Identify some minimal element a^* in $\preceq$.
② Turn a^* into a minimum element by adding constraints (the thick, dark lines).
③ Inductively, find a total ordering of the remaining partial order (the shaded box).

Figure 8.28 A sketch of the proof of Theorem 8.23.

Theorem 8.23: Extending any partial order to a total order.
Let A be a finite set with a partial order $\preceq$. Then there is a total order $\preceq_{\text{total}}$ on A that's consistent with $\preceq$.

Proof. We'll proceed by induction on $|A|$.

For the base case ($|A| = 1$), the task is trivial: there's simply nothing to do! The relation $\preceq$ must be $\{\langle a, a\rangle\}$, where $A = \{a\}$, because partial orders are reflexive. And the relation $\{\langle a, a\rangle\}$ *is* a total order on $\{a\}$ that's consistent with $\preceq$.

For the inductive case ($|A| \geq 2$), we assume the inductive hypothesis (for any set A' of size $|A'| = |A| - 1$ and any partial order on A', there's a total order on A' consistent with that partial order). We must show how to extend $\preceq$ to be a total order on all of A. Here's the idea: we'll remove some element of A that can go first in the total order, inductively find a total order of all the remaining elements, and then add the removed element to the beginning of the order.

More specifically, let $a^* \in A$ be an arbitrary minimal element under $\preceq$ on A—in other words, let a^* be any element such that no $b \in A - \{a^*\}$ satisfies $b \preceq a^*$. Such an element is guaranteed to exist by Theorem 8.21. Add any missing pair $\langle a^*, b\rangle$ to $\preceq$. After the additions, the relation $\preceq$ is still a partial order on A: by the definition of a minimal element, we haven't introduced any violations of transitivity or antisymmetry. Now, inductively, we extend the partial order $\preceq$ on $A - \{a^*\}$ to a total order; the result is a total order on A that's consistent with $\preceq$. (See Figure 8.28.)

(Slightly more formally: define $\preceq'$ as the relation $\preceq \cap\, (A - \{a^*\}) \times (A - \{a^*\})$, which is $\preceq$ restricted to $A - \{a^*\}$. Then $\preceq'$ is a partial order on $A - \{a^*\}$; by the inductive hypothesis, there exists a total order

$\preceq'_{\text{total}}$ on $A - \{a^*\}$ consistent with $\preceq'$. Define

$$\preceq_{\text{total}} = \preceq'_{\text{total}} \cup \{\langle a^*, y\rangle : y \in A\}\,.$$

It's not too hard to verify that $\preceq_{\text{total}}$ is a total order on A that's consistent with $\preceq$.) $\square$

Taking it further: Deciding the order in which to compute the cells of a spreadsheet (where a cell might depend on a list of other cells' contents) is solved using a topological ordering. In this setting, let C denote the set of cells in the spreadsheet, and define a relation $R \subseteq C \times C$ where $\langle c, c'\rangle \in R$ if we need to know the value in cell c before we can compute the value for c'. (For example, if cell `C4`'s value is determined by the formula `A1 + B1 + C1`, then the three pairs $\langle$`A1`, `C4`$\rangle$, $\langle$`B1`, `C4`$\rangle$, and $\langle$`C1`, `C4`$\rangle$ are all in R. Note that it's not possible to compute all the values in a spreadsheet if there's a cell x whose value depends on cell y, which depends on $\cdots$, which depends on cell x—in other words, the "depends on" relationship cannot have a cycle! Furthermore, we're in trouble if there's a cell x whose value depends on x itself. In other words, we can compute the values in a spreadsheet if and only if R is irreflexive and transitive—that is, if R is a strict partial order.

Another problem that can be solved using the idea of topological ordering is that of *hidden-surface removal* in computer graphics: we have a 3-dimensional "scene" of objects that we'd like to display on a 2-dimensional screen, as if it were being viewed from a camera. We need to figure out which of the objects are invisible from the camera (and therefore need not be drawn) because they're "behind" other objects. One classic algorithm, called the *painter's algorithm,* solves this problem using ideas from relations and topological ordering. See p. 8-61.

Computer Science Connections

Deterministic Finite Automata (DFAs)

As we hinted at previously (see the discussion of regular expressions on p. 8-40), there are some interesting computational applications of *finite-state machines,* a formal model for a computational device that uses a fixed amount of memory to respond to input. Variations on these machines can be used in building very simple characters in a video game, in computer architecture, in software systems to do automatic speech recognition, and other tasks. They can also identify which strings match a given regular expression—in fact, for a set of strings L, there exists a finite-state machine M that recognizes precisely the strings in L if and only if there's a regular expression α that matches precisely the strings in L. Formally, a *deterministic finite automaton (DFA)*—the simplest version of a finite-state machine—is defined by five things:

$\Sigma = \{0, 1\}$

$Q = \{a, b, c, win, lose\}$

δ is defined by this table:

	0	1
a	b	c
b	win	$lose$
c	$lose$	win
win	win	win
$lose$	$lose$	$lose$

If you're currently in state a and the input symbol is a 1, then move to state c.

the start state is a

win is the (only) final state

The *start state* is marked with an unattached incoming arrow; from state q on input symbol a, the arrow leaving q with label a points to $\delta(q, a)$. Final states are circled.

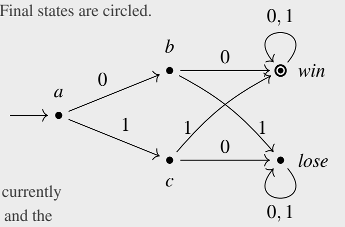
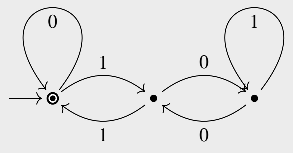


Figure 8.29 A DFA accepting all bitstrings whose first two symbols are the same—both by defining all five components, and by a picture.

- a finite set Σ (the *alphabet*) that defines the set of input symbols the machine can handle;
- a finite set Q (the *states*); the machine is always in one of these states. (The fact that Q is finite corresponds to M having only finite memory.)
- a function $\delta : Q \times \Sigma \to Q$ (the *transition function*): when the machine is in state $q \in Q$ and sees an input symbol $a \in \Sigma$, the machine moves into state $\delta(q, a)$.
- a *start state* $s \in Q$, the state in which the machine begins before having seen any input.
- a set $F \subseteq Q$ of *final states.* If, after processing a string x, the machine ends up in a state $q \in F$, then the machine *accepts* x; if it ends in a state $q \notin F$, then the machine *rejects* x.

An example of a DFA that accepts all bitstrings whose first two symbols are the same is shown in Figure 8.29.

We can also understand DFAs—and the sorts of sets of strings that they can recognize—by thinking about equivalence relations. To see this connection, suppose we're trying to identify binary strings representing integers that are multiples of 3. (So 11 and 1001 and 1111 are all "yes" because $3 \mid 3$ and $3 \mid 9$ and $3 \mid 15$, but 10001 is "no" because $3 \nmid 17$.) Here's one way to solve this problem. Let's define an equivalence relation on binary strings, where $x \equiv y$ if and only if, for any bitstring z, we have that (xz is divisible by 3) $\Leftrightarrow$ (yz is divisible by 3). In other words, two bitstrings x and y are equivalent if, no matter what additional bitstring suffix we add to both of them, the two resulting bitstrings are either both divisible by three or both not divisible by three. For example, it turns out that $11 \equiv 1001$ (11 and 1001 are both 'yes'; 11$\underline{0}$ and 1001$\underline{0}$ are both 'yes'; 11$\underline{1}$ and 1001$\underline{1}$ are both 'no'; 11$\underline{10}$ and



The input is divisible by three if and only if we end up in the leftmost state.

Figure 8.30 A DFA for bitstrings representing multiples of 3.

$1000\underline{10}$ are both 'no'; etc.). Similarly, we have $1000 \equiv 10$. It's not hard to prove that $\equiv$ is an equivalence relation. It's also true, though a bit harder to prove, that there are only three equivalence classes for $\equiv$. (Those equivalence classes are: bitstrings that are 0 mod 3, those that are 1 mod 3, and those that are 2 mod 3.) Thus we can actually figure out whether a bitstring is evenly divisible by 3 with the simple DFA in Figure 8.30. The three states of this machine, going from left to right, correspond to the three equivalence classes for $\equiv$—namely $[0]$, $[1]$, and $[10]$. (For a set of strings that cannot be recognized by a DFA—for example, bitstrings with an equal number of 0s and 1s—there are an infinite number of equivalence classes for $\equiv$.)

(These particular DFAs merely hint at the kind of problem that can be solved with this kind of machine—for much more, see any good textbook in formal languages, such as [73] or [120].)

Computer Science Connections

The Painter's Algorithm and Hidden-Surface Removal

At a high level, the goal in computer graphics is to take a 3-dimensional scene—a set of objects in $\mathbb{R}^3$ (with differing shapes, colors, surface reflectivities, textures, etc.)—as seen from a particular vantage point (a point and a direction, also in $\mathbb{R}^3$). The task is then to *project* the scene into a 2-dimensional image. There are a lot of components to this task, and we've already talked a bit about some of them: typically we'll approximate the shapes of the objects using a large collection of triangles (see p. 5-36), and then compute where each triangle shows up in the camera's view, in $\mathbb{R}^2$, via rotation (see p. 2-63). Even after triangulation and rotation, we are still left with another important step: when two triangles overlap in the 2-dimensional image, we have to figure out which to draw—that is, which one is obscured by the other. This task is also known *hidden-surface removal:* we want to omit whatever pieces of the image aren't visible. For example, to render the humble forest scene in Figure 8.31, we have to draw trees in front of and behind the house, and one particular tree in front of another. One approach to hidden-surface removal is called the *Painter's Algorithm,* named after a hypothetical artist at an easel: we can "paint" the shapes in the image "from back to front," simply painting over faraway shapes with the closer ones as we go, as in Figure 8.32.

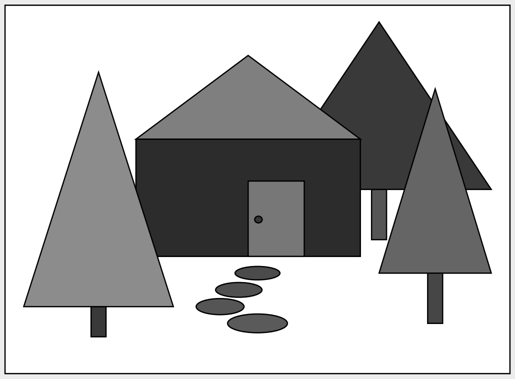


Figure 8.31 A house in a forest.

How might we implement this approach? Let S be the set of shapes that we have to draw. We can compute a relation *obscures* $\subseteq S \times S$, where a pair $\langle s_1, s_2 \rangle$ in *obscures* tells us that we must draw s_2 before s_1. We seek a total order on S that is consistent with the *obscures* relation; we'll draw the shapes in this order.

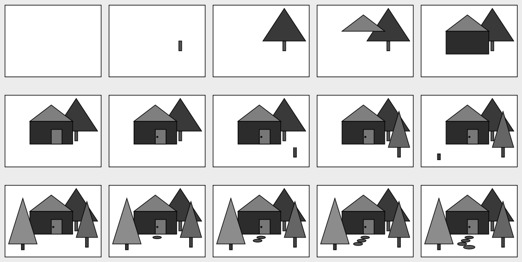


Figure 8.32 Drawing the house, one piece at a time.

Unfortunately *obscures* may not be a total order—or even a partial order! The biggest problem with *obscures* is that we can have "cycles of obscurity"—s_1 obscures s_2 which obscures s_3 which, eventually, obscures a shape s_k that obscures s_1. (See Figure 8.33; although it may look like an M. C. Escher drawing, there's nothing strange going on—just three triangles that overlap a bit like a pretzel.) This issue can be resolved using some geometric algorithms specific to the particular task: we'll *split up* shapes in each cycle of obscurity—here, dividing one triangle into two—so that we no longer have any cycles. (Again see Figure 8.33.)

We now have an expanded set S' of shapes, and a cycle-free relation *obscures* on S'. We can use this relation to compute the order in which to draw the

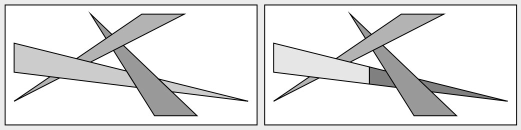


Figure 8.33 A cycle of obscurity, and splitting one of the cycle's pieces to break the cycle.

shapes. First, compute the reflexive, transitive closure of *obscures* on S'. (The resulting relation is a partial order on S'.) Then, extend this partial order to a total order on S', using Theorem 8.23. We now have a total ordering on the shapes that respect the *obscures* relation, so we can draw the shapes in precisely this order. (While the Painter's Algorithm does correctly accomplish hidden-surface removal, it's pretty slow (particularly as we've described it here). For example, when there are many layers to a scene, we actually have to "paint" each pixel in the resulting image many many times. Every computation of a pixel's color before the last is a waste of time. You can learn about cleverer approaches to hidden-surface removal, like the "z-buffer," in a good textbook on computer graphics, such as [61].)

EXERCISES

8.109 List all equivalence relations on $\{0,1\}$.

8.110 List all equivalence relations on $\{0,1,2,3\}$.

Are the following relations on $\mathscr{P}(\{0,1,2,3\})$ equivalence relations? If so, list the equivalence classes under the relation; if not, explain why not.

8.111 $\langle A,B\rangle \in R_1$ if and only if (i) A and B are nonempty and the largest element in A equals the largest element in B, or (ii) if $A = B = \varnothing$.

8.112 $\langle A,B\rangle \in R_2$ if and only if the sum of the elements in A equals the sum of the elements in B.

8.113 $\langle A,B\rangle \in R_3$ if and only if the sum of the elements in A equals the sum of the elements in B and the largest element in A equals the largest element in B. (That is, $R_3 = R_1 \cap R_2$.)

8.114 $\langle A,B\rangle \in R_4$ if and only $A \cap B \neq \varnothing$.

8.115 $\langle A,B\rangle \in R_5$ if and only $|A| = |B|$.

In Example 8.11, we considered the relation $M = \{\langle m,d\rangle : \text{in some years, month } m \text{ has } d \text{ days}\}$, and computed the pairs in the relation $M^{-1}\circ M$. By checking all the requirements (or by visual inspection of Figure 8.10b), we see that $M^{-1}\circ M$ is an equivalence relation. But it turns out that the fact that $M^{-1}\circ M$ is an equivalence relation says something particular about M, and is not true in general. Let $R \subseteq A \times B$ be an arbitrary relation. Which of the three required properties of an equivalence relation must $R^{-1}\circ R$ have? (At least one of these is false!).

8.116 Prove or disprove: $R^{-1}\circ R$ must be reflexive.

8.117 Prove or disprove: $R^{-1}\circ R$ must be symmetric.

8.118 Prove or disprove: $R^{-1}\circ R$ must be transitive.

Let A be any set. There exist two equivalence relations $\equiv_{coarsest}$ and $\equiv_{finest}$ with the following property: if $\equiv$ is an equivalence relation on A, then (i) $\equiv$ refines $\equiv_{coarsest}$, and (ii) $\equiv_{finest}$ refines $\equiv$.

8.119 Identify $\equiv_{\text{coarsest}}$, prove that it's an equivalence relation, and prove property (i): if $\equiv$ is an equivalence relation on A, then $\equiv$ refines $\equiv_{\text{coarsest}}$.

8.120 Identify $\equiv_{\text{finest}}$, prove that it's an equivalence relation, and prove property (ii): if $\equiv$ is an equivalence relation on A, then $\equiv_{\text{finest}}$ refines $\equiv$.

8.121 Write $\equiv_k$ to denote equivalence mod k—that is, $a \equiv_k b$ if and only if $a \bmod k = b \bmod k$. Consider the equivalence relation $\equiv_{60}$. For what values of k is $\equiv_k$ a coarsening of $\equiv_{60}$? For what values of k is $\equiv_k$ a refinement of $\equiv_{60}$?

8.122 Suppose that R is an equivalence relation that coarsens $\equiv_{60}$. Prove or disprove: R is $\equiv_k$, for some integer k.

8.123 In many programming languages, there are two distinct but related notions of "equality": *has the same value as* and *is the same object as*. In Python, these are denoted as `==` and `is`, respectively; in Java, they are `.equals()` and `==`, respectively. (Confusingly!) (For example, in Python, `[1,7,8] + [9] is [1,7,8,9]` is false, but `[1,7,8] + [9] == [1,7,8,9]` is true.) Does one of these equality relations refine the other? Explain.

8.124 List all partial orders on $\{0,1\}$.

8.125 List all partial orders on $\{0,1,2\}$.

Are the following relations on $\mathscr{P}(\{2,3,4,5\})$ partial orders, strict partial orders, or neither? Explain.

8.126 $\langle A,B\rangle \in R_1 \Leftrightarrow \sum_{a\in A} a \leq \sum_{b\in B} b$

8.127 $\langle A,B\rangle \in R_2 \Leftrightarrow \prod_{a\in A} a \leq \prod_{b\in B} b$

8.128 $\langle A,B\rangle \in R_3 \Leftrightarrow A \subseteq B$

8.129 $\langle A,B\rangle \in R_4 \Leftrightarrow A \supseteq B$

8.130 $\langle A, B\rangle \in R_5 \Leftrightarrow |A| < |B|$

8.131 Prove that $\preceq$ is a partial order if and only if $\preceq^{-1}$ is a partial order.

8.132 Prove that if $\preceq$ is a partial order, then $\{\langle a, b\rangle : a \preceq b \text{ and } a \neq b\}$ is a strict partial order.

8.133 A *cycle* in a relation R is a sequence of k distinct elements $a_0, a_1, \ldots, a_{k-1} \in A$ where $\langle a_i, a_{i+1 \bmod k}\rangle \in R$ for every $i \in \{0, 1, \ldots, k-1\}$. A cycle is *nontrivial* if $k \geq 2$. Prove that there are no nontrivial cycles in any transitive, antisymmetric relation R. (Hint: use induction on the length k of the cycle.)

Let $S \in \mathbb{Z}^{\geq 1} \times \mathbb{Z}^{\geq 1}$ be a collection of points. Define the relation $R \subseteq S \times S$ as follows: $\langle\langle a, b\rangle, \langle x, y\rangle\rangle \in R$ if and only if $a \leq x$ and $b \leq y$. (You can think of $\langle a, b\rangle \in S$ as an a-by-b picture frame, and $\langle f, f'\rangle \in R$ if and only if f fits inside f'. Or you can think of $\langle a, b\rangle \in S$ as a job that you'd get a "happiness points" from doing and that pays you b dollars, and $\langle j, j'\rangle \in R$ if and only if j generates no more happiness and pays no more than j'.

8.134 Show that R might not be a total order by identifying two incomparable elements of $\mathbb{Z}^{\geq 1} \times \mathbb{Z}^{\geq 1}$.

8.135 Prove that R must be a partial order.

8.136 Write out all pairs in the relation represented by the Hasse diagram in Figure 8.34a.

8.137 Repeat for Figure 8.34b.

8.138 Repeat for Figure 8.34c.

8.139 Repeat for Figure 8.34d.

8.140 Draw the Hasse diagram for the partial order $\subseteq$ on the set $\mathscr{P}(1, 2, 3)$.

8.141 Draw the Hasse diagram for the partial order $\preceq$ on the set $S = \{0, 1\} \cup \{0, 1\}^2 \cup \{0, 1\}^3$, where, for two bitstrings $x, y \in S$, we have $x \preceq y$ if and only if x is a prefix of y.

Let $\preceq$ be a partial order on A. Recall that an immediate successor *of $a \in A$ is an element c such that (i) $a \preceq c$, and (ii) there is no $b \notin \{a, c\}$ such that $a \preceq b$ and $b \preceq c$. In this case a is said to be an* immediate predecessor *of c.*

8.142 For the partial order $\geq$ on $\mathbb{Z}^{\geq 1}$, identify all the immediate predecessor(s) and immediate successor(s) of 202.

8.143 For the partial order $|$ (divides) on $\mathbb{Z}^{\geq 1}$, identify all the immediate predecessor(s) and immediate successor(s) of 202.

8.144 Give an example of a strict partial order on $\mathbb{Z}^{\geq 1}$ such that *every* integer has exactly two different immediate successors.

8.145 Prove that for a partial order $\preceq$ on A *when A is finite* there must be an $a \in A$ that has fewer than two immediate successors.

8.146 Consider the partial order $\geq$ on the set $\mathbb{Z}^{\geq 0}$. Argue that there is *no* maximal element in $\mathbb{Z}$.

8.147 Note that there *is* a minimal element under the partial order $\geq$ on $\mathbb{Z}^{\geq 0}$—namely 0, which is also the minimum element. Give an example of a partial order on an infinite set that has *neither* a minimal *nor* a maximal element.

8.148 Let $\preceq$ be a partial order on a set A. Prove that there is at most one minimum element in A under $\preceq$. (That is, prove that if $a \in A$ and $b \in A$ are both minimum elements, then $a = b$.)

8.149 Let $\preceq$ be a partial order on a set A, and let $a \in A$ be a minimum element under $\preceq$. Prove that a is also a minim*al* element.

Here's a (surprisingly addictive) word game that can be played with a set of Scrabble tiles. Each player has a set of words that she "owns"; there is also a set of individual tiles in the middle of the table. At any moment, a player can form a new word by taking both (1) one or more tiles from the middle, and (2) zero or more words owned by any of the players; and reordering those letters to form a new word, which the player now owns. For example, from the word `GRAMPS` and the letters `R` and `O`, a player could make the word `PROGRAMS`.

(If you're bored and decide to waste time playing this game: it's more fun if you forbid stealing words with "trivial" changes, like changing `COMPUTER` into `COMPUTERS`. Each player should also get a fair share of the tiles, originally face down; anyone can flip a new tile into the middle of the table at any time.)

Define a relation $\preceq$ on the set W of English words (of three or more letters), as follows: $w \preceq w'$ if w' can be formed from word w plus one or more individual letters. For example, we showed above that `GRAMPS` $\preceq$ `PROGRAMS`.

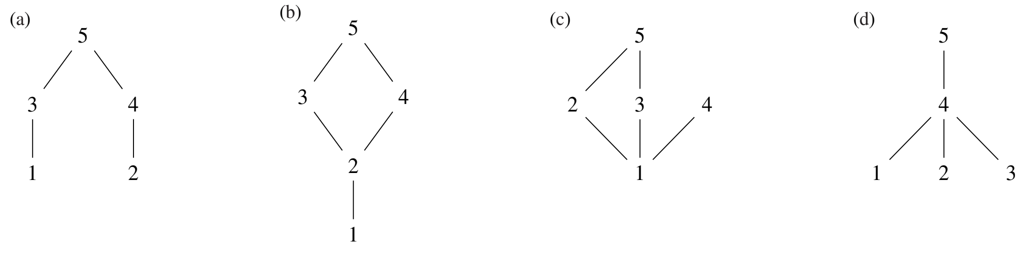


Figure 8.34 Four Hasse diagrams.

8.150 Give a description (in English) of what it means for a word w to be a minimal element under $\preceq$, and what it means for a word w' to be a maximal element under $\preceq$.

8.151 *(programming required.)* Write a program that, given a word w, finds all immediate successors of w. (You can find a dictionary of English words on the web, or `/usr/share/dict/words` on Unix-based operating systems.) Report all immediate successors of `GRAMPS` using your dictionary.

8.152 *(programming required.)* Write a program to find the English word that is the *longest* minimal element under $\preceq$ (that is, out of all minimal elements, find the one that contains the most letters).

8.153 Consider a spreadsheet containing a set of cells C. A cell c can contain a *formula* that depends on zero or more other cells. Write $\preceq$ to denote the relation $\{\langle p, s\rangle : \text{cell } s \text{ depends on cell } p\}$. For example, the value of cell `C2` might be the result of the formula `A2` $*$ `B1`; here `A2` $\preceq$ `C2` and `B1` $\preceq$ `C2`. A spreadsheet is only meaningful if $\preceq$ is a strict partial order. Give a description (in English) of what it means for a cell c to be a minimal element under $\preceq$, and what it means for a cell c' to be a maximal element under $\preceq$.

8.154 List all total orders consistent with the partial order in Figure 8.34a.

8.155 Repeat for the partial order in Figure 8.34b.

8.156 Repeat for the partial order in Figure 8.34c.

8.157 Repeat for the partial order in Figure 8.34d.

A chain *in a partial order $\preceq$ on A is a set $C \subseteq A$ such that $\preceq$ imposes a total order on C—that is, writing the elements of C as $C = \{c_1, c_2, \ldots, c_k\}$ [in an appropriate order], we have $c_1 \preceq c_2 \preceq \cdots \preceq c_k$.*

8.158 Identify all chains of $k \geq 2$ elements in the partial order in Figure 8.34a.

8.159 Repeat for the partial order reproduced in Figure 8.34b.

An antichain *in a partial order $\preceq$ on A is a set $S \subseteq A$ such that no two distinct elements in S are comparable under $\preceq$—that is, for any distinct $a, b \in S$ we have $a \not\preceq b$.*

8.160 Identify all antichains S with $|S| \geq 2$ in the partial order in Figure 8.34a.

8.161 Repeat for the partial order reproduced in Figure 8.34b.

8.162 Consider the set $A = \{1, 2, \ldots, n\}$. Consider the following claim: *there exists a relation $\preceq$ on the set A that is* both *an equivalence relation* and *a partial order.* Either prove that the claim is true (and describe, as precisely as possible, the structure of any such relation $\preceq$) or disprove the claim.

8.5 Chapter at a Glance

Formal Introduction

A *(binary) relation on* $A \times B$ is a subset of $A \times B$. For a relation R on $A \times B$, we can write $\langle a, b \rangle \in R$ or $a\,R\,b$. When A and B are both finite, we can describe R using a two-column table, where a row containing a and b corresponds to $\langle a, b \rangle \in R$. Or we can view R graphically: draw all elements of A in one column, all elements of B in a second column, and draw a line connecting $a \in A$ to $b \in B$ whenever $\langle a, b \rangle \in R$.

We'll frequently be interested in a relation that's a subset of $A \times A$, where the two sets are the same. In this case, we may refer to a subset of $A \times A$ as simply a *relation on* A. For a relation $R \subseteq A \times A$, it's more convenient to visualize R using a *directed graph,* without separated columns: we simply draw each element of A, with an arrow from a_1 to a_2 whenever $\langle a_1, a_2 \rangle \in R$.

The *inverse* of a relation $R \subseteq A \times B$ is a new relation, denoted R^{-1}, that "flips around" every pair in R, so that the relation $R^{-1} = \{\langle b, a \rangle : \langle a, b \rangle \in R\}$ is a subset of $B \times A$. The *composition* of two relations $R \subseteq A \times B$ and $S \subseteq B \times C$ is a new relation, denoted $S \circ R$, that, informally, represents the successive "application" of R and S. A pair $\langle a, c \rangle$ is related under $S \circ R \subseteq A \times C$ if and only if there exists an element $b \in B$ such that $\langle a, b \rangle \in R$ and $\langle b, c \rangle \in S$.

For sets A and B, a *function f from A to B*, written $f : A \to B$, is a special kind of relation on $A \times B$ where, for every $a \in A$, there exists one and only one element $b \in B$ such that $\langle a, b \rangle \in f$.

An *n-ary relation* is a generalization of a binary relation ($n = 2$) to describe a relationship among n-tuples, rather than just pairs. An n-ary relation on the set $A_1 \times A_2 \times \cdots \times A_n$ is just a subset of $A_1 \times A_2 \times \cdots \times A_n$; an n-ary relation on a set A is a subset of A^n.

Properties of Relations: Reflexivity, Symmetry, and Transitivity

A relation R on A is *reflexive* if, for every $a \in A$, we have that $\langle a, a \rangle \in R$. It's *irreflexive* if $\langle a, a \rangle \notin R$ for every $a \in A$. (In the visualization described above, where we draw an arrow $a_1 \to a_2$ whenever $\langle a_1, a_2 \rangle \in R$, reflexivity corresponds to every element having a "self-loop" and irreflexivity corresponds to no self-loops.) Note that a relation might be *neither* reflexive nor irreflexive.

A relation R on A is *symmetric* if, for every $a, b \in A$, we have $\langle a, b \rangle \in R$ if and only if $\langle b, a \rangle \in R$. The relation is *antisymmetric* if the only time both $\langle a, b \rangle \in R$ and $\langle b, a \rangle \in R$ is when $a = b$, and it's *asymmetric* if it's never the case that $\langle a, b \rangle \in R$ and $\langle b, a \rangle \in R$ whether $a \neq b$ or $a = b$. Note that, while asymmetry implies antisymmetry, they are different properties—and they're both different from "not symmetric"; a relation might not be symmetric, antisymmetric, *or* asymmetric. (In the visualization, a relation is symmetric if every arrow $a \to b$ is matched by an arrow $b \to a$; it's antisymmetric if there are no matched bidirectional pairs of arrows between a and $b \neq a$; and it's asymmetric if it's antisymmetric and furthermore there aren't even any self-loops.) An alternative view is that a relation R is symmetric if and only if $R \cap R^{-1} = R = R^{-1}$; it's antisymmetric if and only if $R \cap R^{-1} \subseteq \{\langle a,a \rangle : a \in A\}$; and it's asymmetric if and only if $R \cap R^{-1} = \varnothing$.

A relation R on A is *transitive* if, for every $a, b, c \in A$, if $\langle a,b \rangle \in R$ and $\langle b,c \rangle \in R$, then $\langle a,c \rangle \in R$ too. In the visualization, R is transitive if there are no "open triangles": in a chain of connected elements, every element is also connected to all "downstream" connections. The relation R is transitive if and only if $R \circ R \subseteq R$.

For a relation $R \subseteq A \times A$, the *closure* of R with respect to some property is the smallest relation $R' \supseteq R$ that has the named property. For example, the *symmetric closure of* R is the smallest relation $R'' \supseteq R$ such that R'' is symmetric. We also define the *reflexive closure* R'; the *transitive closure* R^+; the *reflexive transitive closure* R^*; and the *reflexive symmetric transitive closure* $R^{\equiv}$. When A is finite, we can compute any of these closures by repeatedly adding any missing elements to the set. The reflexive closure of R is given by $R \cup \{\langle a,a \rangle : a \in A\}$; the symmetric closure of R is $R \cup R^{-1}$; and the transitive closure of R is $R \cup R^2 \cup R^3 \cup \cdots$.

Special Relations: Equivalence Relations and Partial/Total Orders

There are two special kinds of relations that emerge from particular combinations of these properties: *equivalence relations* and *partial/total orders*.

Equivalence relations. An *equivalence relation* is a relation $\equiv$ that's reflexive, symmetric, and transitive. Such a relation partitions the elements of A into one or more categories, called *equivalence classes;* any two elements in the same equivalence class are related by $\equiv$, and no two elements in different equivalence classes are related.

A *refinement* of $\equiv$ is another equivalence relation $\equiv_r$ on the same set A where $a \equiv b$ whenever $a \equiv_r b$. Each equivalence class of $\equiv$ is partitioned into one or more equivalence classes by $\equiv_r$, but no equivalence class of $\equiv_r$ intersects with more than one equivalence class of $\equiv$. We also call $\equiv$ a *coarsening* of $\equiv_r$.

Partial and total orders. A *partial order* is a reflexive, antisymmetric, and transitive relation $\preceq$. (A *strict partial order* $\prec$ is *irreflexive,* antisymmetric, and transitive.) Elements a and b are *comparable under* $\preceq$ if either $a \preceq b$ or $b \preceq a$; otherwise they're *incomparable.* A *Hasse diagram* is a simplified visual representation of a partial order where we draw a physically below c whenever $a \preceq c$, and we omit the $a \rightarrow c$ arrow if there's some other element b such that $a \preceq b \preceq c$. (We also omit self-loops.)

For a partial order $\preceq$ on A, a *minimum element* is an element $a \in A$ such that, for every $b \in A$, we have $a \preceq b$; a *minimal element* is an $a \in A$ such that, for every $b \in A$ with $b \neq a$, we have $b \not\preceq a$. (*Maximum* and *maximal elements* are defined analogously.) Every minimum element is also minimal, but a minimal element a isn't minimum unless a is comparable with every other element. There's at least one minimal element in any partial order on a finite set.

A *total order* is a partial order under which all pairs of elements are comparable. A total order $\preceq_{\text{total}}$ is *consistent with the partial order* $\preceq$ if $a \preceq b$ implies that $a \preceq_{\text{total}} b$. For any partial order $\preceq$ on a finite set A, there is a total order $\preceq_{\text{total}}$ on A that's consistent with $\preceq$. Such an ordering of A is called a *topological ordering* of A.

Key Terms and Results

Key Terms

Formal Introduction

- (binary) relation
- inverse (of a relation)
- composition (of two relations)
- functions (as relations)
- *n*-ary relation

Properties of Relations

- reflexivity
- irreflexivity
- symmetry
- asymmetry
- antisymmetry
- transitivity
- closures (of a relation)

Special Relations

- equivalence relation
- equivalence class
- coarsening, refinement
- partial order
- comparability
- total order
- Hasse diagram
- minimal/maximal element
- minimum/maximum element
- consistency
  (of a total order with a partial order)
- topological ordering

Key Results

Formal Introduction

1 For relations $R \subseteq A \times B$ and $S \subseteq B \times C$, the relations $R^{-1} \subseteq B \times A$ and $S \circ R \subseteq A \times C$—the inverse of R and the composition of R and S—are defined as

$$\begin{aligned} R^{-1} &= \{\langle b,a\rangle : \langle a,b\rangle \in R\} \\ S \circ R &= \{\langle a,c\rangle : \\ &\qquad \exists b \in B \text{ such that } \langle a,b\rangle \in R \text{ and } \langle b,c\rangle \in S\}. \end{aligned}$$

2 A function $f : A \to B$ is a special case of a relation on $A \times B$, where, for every $a \in A$, there exists one and only one element $b \in B$ such that $\langle a,b\rangle \in f$.

Properties of Relations

1 A relation R is symmetric if and only if $R \cap R^{-1} = R = R^{-1}$; it's antisymmetric if and only if $R \cap R^{-1} \subseteq \{\langle a,a\rangle : a \in A\}$; and it's asymmetric if and only if $R \cap R^{-1} = \varnothing$.

2 A relation R is transitive if and only if $R \circ R \subseteq R$.

3 The reflexive closure of R is $R \cup \{\langle a,a\rangle : a \in A\}$; the symmetric closure of R is $R \cup R^{-1}$; and the transitive closure of R is $R \cup R^2 \cup R^3 \cup \cdots$.

Special Relations

1 For a partial order $\preceq \subseteq A \times A$ on a *finite* set A, there is at least one minimal element and at least one maximal element under $\preceq$.

2 Let A be any finite set with a partial order $\preceq$. Then there is a total order $\preceq_{\text{total}}$ (a *topological ordering* of A) on A that's consistent with $\preceq$.