Relations

1 Relations

A "relation" is a fundamental mathematical notion expressing a relationship between elements of sets.
It's an abstract notion useful in practice for modeling many different sorts of relationships.
It's the basis of the *relational data base* model, the standard data model for practical data processing systems.

1.1 Definitions

A *binary relation from* a set A *to* a set B is a subset $R \subseteq A \times B$.
A binary relation *on* a set A is a subset $R \subseteq A^2$ (I.e., a relation from A to A).
We often write $a \sim_R b$ or aRb instead of $(a, b) \in R$.

We can also define a *ternary* relation on A as a subset $R \subseteq A^3$ or, in general, an n-ary relation as a subset $R \subseteq A^n$, or $R \subseteq A_1 \times A_2 \times \cdots \times A_n$ if the sets A_i are different.

Relations are used to model many different things:

1. The relation "is taking class" as a subset of $\{$students at MIT$\} \times \{$classes at MIT$\}$. A relation from students to classes.
2. The relation "is living in the same room" as a subset of $\{$students at MIT$\} \times \{$students at MIT$\}$. A relation on students.
3. The relation "can drive from first to second city". (Not necessarily directly – just some way, on some roads.)
4. Relation on computers, "are connected (directly) by a wire"
5. "meet one another on a given day"
6. "likes"
7. Let $A = \mathbb{N}$ and define $a \sim_R b$ iff $a \leq b$.
8. Let $A = \mathcal{P}(\mathbb{N})$ and define $a \sim_R b$ iff $|a \cap b|$ is finite.
9. Let $A = \Re^2$ and define $a \sim_R b$ iff $d(a, b) = 1$.
10. Let $A = \mathcal{P}(\{1, \cdots, n\})$ and let $a \sim_R b$ if $a \subseteq b$.

1.2 Properties of Relations

There are several standard properties that are satisfied by (some) relations.
These occur commonly enough that they are worth identifying.
Sometimes interesting things can be deduced just from the abstract properties.

A binary relation R on A is:

1. *reflexive* if for every $a \in A$, $a \sim_R a$.
2. *irreflexive* if for every $a \in A$, $a \not\sim_R a$.
3. *symmetric* if for every $a, b \in A$, $a \sim_R b$ implies $b \sim_R a$.
4. *antisymmetric* if for every $a, b \in A$, $a \sim_R b$ and $b \sim_R a$ implies $a = b$.
5. *transitive* if for every $a, b, c \in A$, $a \sim_R b$ and $b \sim_R c$ implies $a \sim_R c$.
6. *asymmetric* if for every $a, b \in A$, $a \sim_R b$ implies $\neg(b \sim_R a)$.

There's no sense to the specific negative prefixes; they just need to be remembered.
The difference between antisymmetric and asymmetric relations, is that antisymmetric relations may contain pairs (a, a), i.e., elements can be in relations with themselves, while in an asymmetric relation this is not allowed.
Clearly, any asymmetric relation is also antisymmetric, but not vice versa.

Among our examples:

- Relation 2 is reflexive, symmetric, transitive.
- Relation 3 is reflexive, transitive.
  Not necessarily symmetric, since roads could be one-way (consider Boston), but in actuality....
  But definitely not antisymmetric.
- Relation 4 is symmetric but not transitive.
  Whether it is reflexive is open to interpretation.
- Relation 5 likewise.
- Relation 6 is (unfortunately) not symmetric. Not antisymmetric. Not transitive. Not even reflexive!
- Relation 7 is reflexive, antisymmetric, transitive.
- Relation 8 is not reflexive. It is symmetric. Not transitive.
  Evens$\cap$Odds is finite (empty), but not Evens$\cap$Evens.
- Relation 9 is only symmetric.
- Relation 10 is reflexive, antisymmetric and transitive.

1.3 Representation

There are different ways of representing relations, e.g., for handling in a computer program.

We can describe by properties, as we did above.
That's about all we can do, for infinite sets.

But for finite sets, usually we use some method that explicitly enumerates all the elements of the relation.

Some alternatives: Lists, Matrices, Graphs.

1.3.1 Lists

To represent a finite relation from set A to set B, just list all the pairs.

Ex: Consider for example the relation from
$A = \{0, 1, 2, 3\}$ to $\{a, b, c\}$
defined by the list $\{(0, a), (0, c), (1, c), (2, b), (1, a)\}$.

Ex: The divisibility relation on natural numbers $\{1, \cdots, 12\}$ is represented by the list:
$\{(1, 1), (1, 2), \ldots, (1, 12), (2, 2), (2, 4), \ldots, (2, 12),$
$(3, 3), (3, 6), (3, 9), (3, 12), (4, 4), (4, 8), (4, 12), (5, 5), (6, 6),$
$(6, 12), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11), (12, 12)\}$.

Reflexivity in this representation: Contains all pairs (a, a).
Symmetry: If contains (a, b) then contains (b, a).
Transitivity: If contains (a, b) and (b, c) then contains (a, c).

1.3.2 Boolean matrices

Used for finite sets A, B
Rows for elements of A, columns for B.
1 in position if the pair is in the relation, 0 otherwise.

Ex: The relation from $\{0, 1, 2, 3\}$ to $\{a, b, c\}$ of our first example is represented by the matrix

	a	b	c
0	1	0	1
1	1	0	1
2	0	1	0
3	0	0	0

Ex: The divisibility relation over $\{1, 2, \ldots, 12\}$ is represented by the matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	1	0	1	0	1	0	1	0	1
3	0	0	1	0	0	1	0	0	1	0	0	1
4	0	0	0	1	0	0	0	1	0	0	0	1
5	0	0	0	0	1	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1

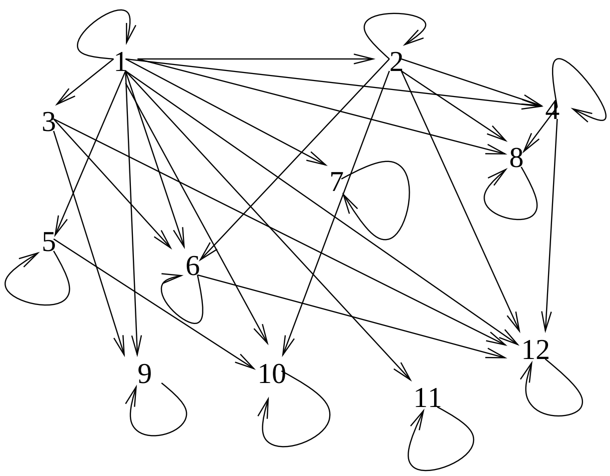
Reflexivity in this representation: major diagonal is all 1
Symmetry: the matrix is symmetric around major diagonal
Transitivity: ...

1.3.3 Digraphs

Digraphs are used for relations where $A = B$, i.e., for relations on a finite set A. To represent a relation as a digraph we draw a dot (vertex) for each element of A, and draw an arrow from first element to second element of each pair in the relation.
The digraph may contain self-loops, i.e. arrows from a dot to itself, associated to the elements a such that (a, a) is in the relation.

The relation from $\{0, 1, 2, 3\}$ to $\{a, b, c\}$ of the previous examples cannot be represented as a digraph because $A \neq B$.

The divisibility relation over $\{1, 2, \ldots, 12\}$ is represented by the digraph



Reflexivity: All nodes have self-loops.
Symmetry: all edges are bidirectional.
Transitivity: Short-circuits, for any sequence of consecutive arrows, there is a single arrow from the first to the last node.

1.4 Paths

Definition 1.1 A *path* in a relation R is a sequence $a_0, \dots, a_k$ with $k \geq 0$ such that $(a_i, a_{i+1}) \in R$ for every $i < k$. We call k the *length* of the path.

In the graph model, a path is something you can trace out by following arrows from vertex to vertex, without lifting your pen.
Note that a singleton vertex is a length 0 path (this is just for convenience).

Definition 1.2 A *simple path* is a path with no repeated vertices.

2 Operations on Relations

2.1 Inverse

If R is a relation on $A \times B$, then R^{-1} is a relation on $B \times A$ given by $R^{-1} = \{(b, a) \mid (a, b) \in R\}$. It's just the relation "turned backwards."

Inverse of "parent-of" is "child-of."

2.2 Composition

The *composition* of relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ is the relation $R_2 \circ R_1 = \{(a, c) \mid (\exists b)((a, b) \in R_1) \wedge ((b, c) \in R_2)\}$.
Notice that $R_2 \circ R_1$ and $R_1 \circ R_2$ are different. (Some prefer to use $R_1 \circ R_2$ for the above composition.)

Examples:
Composition of parent-of relation with itself gives grandparent-of.
Composition of child-of relation with parent-of relation gives sibling-of relation.
Does composition of parent-of with child-of give married-to/domestic partners? No, because misses childless couples.

Composition of relations is equivalent to matrix multiplication, with $+$ replaced by $\vee$ (Boolean or) and $*$ replaced by $\wedge$.

Ex:
Let R_1 be relation from first example above:

	a	b	c
0	1	0	1
1	1	0	1
2	0	1	0
3	0	0	0

Let R_2 be relation from $\{a, b, c\}$ to $\{d, e, f\}$ given by:

	d	e	f
a	1	1	1
b	0	1	0
c	0	0	1

Then $R_2 \circ R_1 = \{(0, d), ...\}$, that is,

	d	e	f
0	1	1	1
1	1	1	1
2	0	1	0
3	0	0	0

Same as matrix multiply, using Boolean ops and and or.

Relational composition is associative.
Just like matrix multiplication.

Lemma 2.1 $R_1 \circ (R_2 \circ R_3) = (R_1 \circ R_2) \circ R_3$

The composition $R \circ R$ of R with itself is written R^2.
Similarly R^n denotes R composed with itself n times.
Recursive definition:
$R_1 = R$, $R^n = R \circ R^{n-1}$.
Actually, can define $R_0 = \{(a, a) \mid a \in R\}$ as base case.
We'll just use 1 as base case, as book does.

Could just as well have defined as $R^{n-1} \circ R$:

Useful observation:

Lemma 2.2 $R^n = \{(a, b) \mid$ there is a length n path from a to b in $R\}$

Before we start the proof consider the relation R, (a, b), (a, c), (b, d), (d, e) or:

	a	b	c	d	e
a	0	1	1	0	0
b	0	0	0	1	0
c	0	0	0	0	0
d	0	0	0	0	1
e	0	0	0	0	0

R^2 will be

	a	b	c	d	e
a	0	0	0	1	0
b	0	0	0	0	1
c	0	0	0	0	0
d	0	0	0	0	0
e	0	0	0	0	0

Draw a graph and find that the only paths of length two in R are $(a,b),(b,d)$ and $(b,d),(d,e)$. Now the proof:

Proof. By induction.
The base case is clear—paths of length one are edges of R.
Suppose it is true for R^n; let's prove it for R^{n+1}.
Suppose there is a path $a_0,\ldots,a_{n+1}$ in R.
This is a path $a_0,\ldots,a_n$ in R followed by a tuple (a_n,a_{n+1}) of R.
Thus (by induction) we have $(a_0,a_n)\in R^n$ and $(a_n,a_{n+1})\in R$.
Thus (by definition of composition) $(a_0,a_{n+1})\in R^{n+1}$ as claimed.

Am I done? No, I only proved set containment in one direction.
I still need to show that if $(a,b)\in R^n$ there is a path of length n from a to b.
But I'll let you do it: use the induction above, but turn the proof backwards. ■

2.3 Closure

A closure "extends" a relation to satisfy some property. But extends it as little as possible.

Definition 2.3 The *closure* of relation R with respect to property P is the relation S that (i) contains R, (ii) has property P, and (iii) is contained in *any* relation satisfying (i) and (ii). That is, S is the "smallest" relation satisfying (i) and (ii).

There might be no one relation satisfying the definition; in that case the closure is not defined.

Lemma 2.4 The reflexive closure of R is $S = R\cup\{(a,a)\mid a\in A\}$

Proof. It contains R and is reflexive by design.
Furthermore (by definition) any relation satisfying (i) must contain R, and any satisfying (ii) must contain the pairs (a,a), so any relation satisfying both (i) and (ii) must contain S. ■

Ex: the following matrix:

	a	b	c	d	e
a	0	1	1	0	0
b	0	0	0	1	0
c	0	0	0	0	0
d	0	0	0	0	1
e	0	0	0	0	0

Will be changed to:

	a	b	c	d	e
a	1	1	1	0	0
b	0	1	0	1	0
c	0	0	1	0	0
d	0	0	0	1	1
e	0	0	0	0	1

Lemma 2.5 The symmetric closure of R is $S = R \cup R^{-1}$.

Proof. This relation is symmetric and contains R.
It is also the smallest such.
For suppose we have some symmetric relation T with $R \subseteq T$.
Consider $(a, b) \in R$.
Then $(a, b) \in T$ so by symmetry $(b, a) \in T$.
It follows that $R^{-1} \subseteq T$.
So $S = R \cup R^{-1} \subseteq T$. ∎

Ex: the following matrix:

	a	b	c	d	e
a	0	1	1	0	0
b	0	0	0	1	0
c	0	0	0	0	0
d	0	0	0	0	1
e	0	0	0	0	0

Will be changed to:

	a	b	c	d	e
a	0	1	1	0	0
b	1	0	0	1	0
c	1	0	0	0	0
d	0	1	0	0	1
e	0	0	0	1	0

Paths give us the terminology for transitive closure.

Lemma 2.6 The transitive closure of a relation R is that set

$$S = \{(a, b) \mid \text{there is a path from } a \text{ to } b \text{ in } R\}.$$

Proof. First let's show S is transitive.
Suppose $(a, b) \in S$ and $(c, d) \in S$.
This means that there is an (a, b) path and a (b, c) path in R.
If we "concatenate" them (attach the end of the (a, b) path to the start of the (b, c) path, we get an (a, c) path. So $(a, c) \in S$.
So S is transitive.

So consider any transitive relation T containing R.
We have to show it contains S.
Assume for contradiction that $S \not\subseteq T$.
This means that some pair $(a, b) \in S$ but $(a, b) \notin T$.
In other words, there is a path $a_0, \ldots, a_k = b$ in S where $(a_0, a_k) \notin T$.
Call this a "missing" path.

Now we use well ordering (i.e. induction).
We have just claimed that the set of paths in T is nonempty.
So consider a shortest missing path $s_0, \ldots, s_m$.
We will derive a contradiction to this being the shortest missing path.

First suppose $m = 1$. The s_0, s_1 is a path in R, so $(s_0, s_1) \in R$.
But we know T contains R, so $(s_0, s_1) \in T$, a contradiction.

Now suppose $m > 1$.
Then $s_1, \ldots, s_{m-1}$ is a path in R $(m - 1 > 0)$.
But it is shorter than our original "shortest missing" path, so cannot be missing.
Thus $(s_1, s_{m-1}) \in T$.
Also we have $(s_{m-1}, s_m) \in T$ since $R \subseteq T$.
Thus by transitivity of T, $(s_1, s_m) \in T$, a contradiction.

We get a contradiction either way, so our assumption (that $S \not\subseteq T$) is false. This proves the theorem. ■

Wait a minute. Well ordering is applied to sets of *numbers*; we applied it to a set of path! How?
Well, look at the set of "lengths of missing paths".
It is nonempty, so has a smallest element.
There is path that has this length—so it is a shortest path.

Ex: the following matrix:

	a	b	c	d	e
a	0	1	0	0	0
b	0	0	0	1	0
c	0	0	0	0	0
d	0	0	0	0	1
e	0	0	0	0	0

Which can be represented by a digraph:

$a \to b \to d \to e$ and an additional node c

Will be changed to:

	a	b	c	d	e
a	0	1	0	1	1
b	0	0	0	1	1
c	0	0	0	0	0
d	0	0	0	0	1
e	0	0	0	0	0

2.3.1 Computing the Transitive Closure

With reflexive and symmetric closure, it is pretty clear how to actually build them.
But for transitive closure, how do we actually find all the paths we need?

Let's start by finding paths of a given length.
Recall the definition of R^n of R composed with itself n times. We proved that
$R^n = \{(a, b) \mid \text{ there is a length } n \text{ path from } a \text{ to } b \text{ in } R\}$

This means we can write the transitive closure of R as $R \cup R^2 \cup R^3 \cdots$.
Better (since we know how to do composition), but still a problem: there are infinitely many terms!

Lemma 2.7 Suppose A has n elements. If there is a path from a to b, there is a path of length at most n from A to B.

Proof. We'll use well ordering (for a change).
Consider the shortest path $a = a_0, a_1, \dots, a_k$ from a to b (we know one exists, so by well ordering there is a shortest one).
Suppose $k > n$.
Then some element of A appears twice in the list (with more than n list entries, one must be a repeat).
This means the path is at some point circling back to where it was before.
We can cut out this cycle from the path and get a shorter path.
This contradicts that we had a shortest path. So we cannot have $k > n$. ■

So we don't need infinitely many terms. It is enough to take paths of length n, namely $R^1 \cup R^2 \cdots \cup R^n$.

3 Equivalence relations and partitions

Now we consider an important special type of relation, called an i*equivalence relation*.
We give an important special case – modular arithmetic.
We show an application to RSA.

3.1 Definitions

Definition 3.1 An *equivalence relation* is a relation that is reflexive, symmetric and transitive.

For example, the "roommates" relation is an equivalence relation. So is "married to", "same size as", and "on same Ethernet hub".
A trivial example is "=" relation on natural numbers.
The hallmark of equivalence relations is the word "same".
It provides a way to "hide" unimportant differences.

Equivalence relations partition the universe into subsets in a natural way:

Definition 3.2 A *partition* of a set A is a collection of subsets $\{A_1, \cdots, A_k\}$ such that any two of them are disjoint (for any $i \neq j$, $A_i \cap A_j = \emptyset$) and such that their union is A.

That is, every element of A is in exactly one subset.

Fix an equivalence relation R on A.
For an element $a \in A$, let $[a]$ denote the set $\{b \in A \mid a \sim_R b\}$.
We call this set the *equivalence class of a under R*.
We call a a *representative* of $[a]$

Lemma 3.3 The sets $[a]$ for $a \in A$ constitute a partition of A. That is, for every $a, b \in A$, either $[a] = [b]$ or $[a] \cap [b] = \emptyset$.

Proof. Fix $a, b \in A$.
If $[a] = [b]$ then we are done, so suppose not.
Let c be any element that is in one of the sets but not the other.
Without loss of generality we can assume that $c \in [b] - [a]$ (we know that either $c \in [b] - [a]$ or $c \in [a] - [b]$. In the latter case we can simply swap a and b and reduce to the first case.).
Similarly, if $[a] \cap [b] = \emptyset$ then we are done, so suppose not.
Let d be any element in $d \in [a] \cap [b]$.

We will get a contradiction by showing that $a \sim_R c$ and therefore that $c \in [a]$.

First, $a \sim_R d$ because $d \in [a]$ (note that $d = a$ is a possibility but this is ok because R is reflexive).
Second, $d \sim_R b$ and $b \sim_R c$ because both $c, d \in [b]$ and R is symmetric.
This implies, by transitivity, that $d \sim_R c$.
Finally, by transitivity, $a \sim_R c$ because $a \sim_R d$ and $d \sim_R c$. ∎

Conversely, any partition $\{A_1, \cdots, A_k\}$ of A defines an equivalence relation by letting $a \sim_R b$ iff a and b are in the same A_i.
(Note "same".)
This is an equivalence relation:

- Reflexivity: for all a we know $a \in A_i$ for some i, by definition of partition. Clearly a and a are in the same A_i, and $a \sim_R b$.
- Symmetry: Assume $a \sim_R b$, that is a and b are in the same A_i. Also b and a are in the same A_i and therefore $b \sim_R a$.
- Transitivity: Assume $a \sim_R b$ and $b \sim_R c$. By definition of $\sim_R$, a and b are in the same A_i for some i, and b and c are in the same A_j for some j. But by definition of partition b cannot be in two different A_i's. So, it must be $A_i = A_j$ and a and c are in the same A_i, proving $a \sim_R c$.

Therefore, we can look at partitions and at equivalence relations as the same thing.

3.2 Integers modulo m

An equivalence relation on integers (positive, negative and 0):

Definition 3.4 If a and b are integers, then we say $a = b \pmod{m}$ if $m \mid (a - b)$.

An equivalent formulation says that $a = b + km$ for some integer k.

Theorem 3.5 Equality modulo m is an equivalence relation.

Proof. We need to show the relation is reflexive, symmetric, and transitive.

- Reflexive: Clearly $m \mid a - a = 0$, so $a = a \pmod{m}$.
- Symmetric: If $a = b + km$ then $b = a + (-k)m$.
- Transitive: Suppose $a = b \pmod{m}$ and $b = c \pmod{m}$.
  Then $m \mid (a - b)$ and $m \mid (b - c)$.
  So $(a - b) = k_1 m$ and $(b - c) = k_2 m$.
  So $(a - c) = (a - b) + (b - c) = (k_1 + k_2)m$.
  Therefore $m \mid= a - c$.

■

The equivalence class of a is simply the set $[a] = \{b \in \mathbb{Z} \mid a = b \pmod{m}\}$, or $\{km + a \mid k \in \mathbb{Z}\}$.

It turns out that we can extend a lot of standard arithmetic to work modulo m.
In fact, we can define notions of sum and product for the equivalence classes mod m.
For example, we define $[a] + [b]$, given two equivalence classes mod m, to be the equivalence class $[a + b]$.
This is not as obvious as it seems: notice that the result is given in terms of a and b, two selected representative from the equivalence classes, but that it is supposed to apply to the equivalence classes themselves.
To prove that this works, we have to show that it doesn't matter which representatives of the equivalence classes we choose:

Lemma 3.6 If $a = x \pmod{m}$ and $b = y \pmod{m}$ then $(a + b) = (x + y) \pmod{m}$.

Proof. $m \mid (a - x)$ and $m \mid (b - y)$ so $m \mid ((a - x) + (b - y)) = (a + b) - (x + y)$, ■

It follows that if we are interested only the result of the addition modulo m, which is the case, for example, in the RSA cryptosystem, that we can at any time replace a given number with a different number equivalent to it, without changing the value (equivalence class) of the final answer.

The same fact can be proven for multiplication.
This is the basis of the fast exponentiation algorithm presented in the next subsection.

3.3 Fast Exponentiation

Recall that for RSA we wanted to compute the remainder of x^a on division by b—that is, the equivalence class mod b of x^a.

In the section on recursive algorithms, we already saw how to reduce the number of multiplications substantially.
But there is still a problem: x^a is a really big number.
It is bigger than 2^a, so has $2^{2^{128}}$ bits if we use a 128-bit key a.
It will take a long time to write this number down (if we could find a place to fit it).

But notice that we only care about the remainder modulo b. So using modular arithmetic we can solve the large-number problem.
Whenever the number we are computing gets too big, replace it with a smaller representative of the same equivalence class.
What is the smallest representative? The remainder on division by b.
We've argued that switching representatives doesn't change the answer.
But now all our math is done on number smaller than b—i.e. 128-bit numbers.

4 Partial Orders

A particular type of binary relation.

Important in computer science.
Applications to task scheduling,
Database concurrency control,
Logical time in distributed computing

4.1 Definitions

4.1.1 Partial Order Definition

Definition 4.1 A binary relation $R \subseteq A \times A$ is a **partial order** if it is reflexive, transitive, and antisymmetric.

Quick review:

- reflexive: aRa
- transitive: $aRb \wedge bRc \Rightarrow aRc$
- symmetric: $aRb \Rightarrow bRa$
- antisymmetric: $aRb \wedge bRa \Rightarrow a = b$. i.e., $\forall a \neq b, aRb \Rightarrow \neg bRa$. Means NEVER symmetric!

For comparison:
Recall that a relation that is reflexive, transitive, and symmetric is an equivalence relation.

The reflexivity, antisymmetry and transitivity properties are abstract properties that generally describe "ordering" relationships.

So we often write an ordering-style symbol like $\preceq$ instead of just a letter like R, for a partial order relation. This lets us use notation similar to $\leq$. For example, we write $a \prec b$ if $a \preceq b$ and $a \neq b$. Similarly, we write $b \succeq a$ as equivalent to $a \preceq b$.

But this could be misleading – note that $\geq$ is a partial order on natural numbers, as well as $\leq$. If we use the $\preceq$ symbol for $\geq$, things look really funny. In cases like this, better to use R.
A partial order is always defined on some set A. The set together with the partial order is called a "poset":

Definition 4.2 A set A together with a partial order $\preceq$ is called a **poset** $(A, \preceq)$.

Examples of posets:

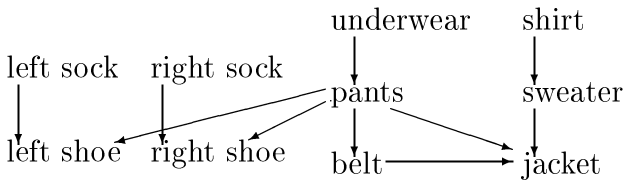
- $A = \mathbb{N}, R = \leq$, easy to check reflexive, transitive, antisymmetric
- $A = \mathbb{N}, R = \geq$, same.
- $A = \mathbb{N}, R = <$, NOT because not reflexive
- $A = \mathbb{N}, R = \mid$ (divides), easy to check reflexive, transitive, antisymmetric
- $A = P(\mathbb{N}), R = \subseteq$, check reflexive: $S \subseteq S$, transitive: $S \subseteq S' \wedge S' \subseteq S'' \Rightarrow S \subseteq S''$, antisymmetric: $S \subseteq S' \wedge S' \subseteq S \Rightarrow S = S'$.
- $A =$ "set of all computers in the world", $R =$ "is (directly or indirectly) connected to".
  NOT a p.o. because it is not true that $aRb \wedge bRa \Rightarrow a = b$.
  In fact, it is symmetric, transitive. Equivalence relation.
- $A =$ "set of all propositions", $R = \Rightarrow$, NOT because not antisymmetric. Not symmetric either, so not equivalence relation.

4.1.2 Directed Acyclic Graphs

A common source of partial orders in computer science is in "task graphs".
You have a set of tasks A, and a relation R in which aRb means "b cannot be done until a is finished".
Implicitly, "if all the things that point at b are done, I can do b."

This can be nicely drawn as a graph.
We draw an arrow from a to b if aRb.

For example, below is a graphs that describes the order in which one would put on clothes.
The set is of clothes, and the edges say what should be put on before what.



The "depends on" graph imposes an ordering on tasks. But is it a partial order?
No, because it isn't reflexive or transitive.
But there is a natural extension: the reflexive transitive closure *is* a partial order.
It gives the relation "must be done before."

Wait, we know the reflexive transitive closure is reflexive and transitive, but is it antisymmetric?
What if I add a relation edge from belt to underwear?
In that case my dependency graph stops making sense: there is no way to get dressed!

What goes wrong? A cyclic dependency.

Definition 4.3 A *cycle* is a path that ends where it started (i.e., the last vertex equals the first).

Definition 4.4 A *directed acyclic graph (DAG)* is a directed graph with no cycles.

Lemma 4.5 Any partial order is a DAG.

Proof. Suppose a partial order R has a cycle $a_1, \ldots, a_k, a_1$.
Then by transitivity of R (with an induction hiding inside) we have $(a_1 R a_k)$. We also have $a_k R a_1$.
This violates the antisymmetry of R, a contradiction. ■

Lemma 4.6 The transitive reflexive closure of a DAG is a partial order.

Proof. Let the DAG be R and its transitive closure S.
We already proved that S is the pairs (a, b) that are connected by a path in R.
S is transitive; we just need to prove it is antisymmetric.
We do so by contradiction. Suppose A is not antisymmetric. Then for some a and b, we have aSb and bSa.
In other words, there is a path from a to b and a path from b to a.
If we attach these two paths, we get a path from a to a.
This contradicts the assumption that R is acyclic. ■

4.1.3 Hasse Diagrams

If we draw the graph of a poset, we may get *lots* of edges (consider the $\leq$ order on some natural numbers).
But a lot of those edges are "implicit" from the transitivity of the order.
We can get a much less messy picture by leaving out these arrows.

Consider the clothing example.
Under the transitive closure, underwear precedes belt.
But we don't need to see that edge to know it is there.
Similarly loops (which express reflexive relationships) aren't shown.

Definition 4.7 A **Hasse diagram** for a poset $(A, \preceq)$ is a graph:
- whose vertices are the elements of A,
- whose edges (arrows) represent pairs (a, b), where $a \leq b$ but $a \neq b$,
- that contains no "transitive edges",
- for which all pairs in $\preceq$ are obtainable by transitivity from edges in the diagram.

It's a "transitive disclosure" of the poset.

To convert the diagram above to an "official form" Hasse diagram, we must remove the pants $\rightarrow$ jacket arrow.

4.1.4 Partial vs. Total Orders

A partial order is called *partial* because it is not necessary that an ordering exist between every pair of elements in the set.

E.g., Lshoe and Rshoe have no prescribed ordering between them.

E.g., For two sets, it's not necessary that either be a subset of the other.

When there is no prescribed order between two elements we say that they are "incomparable".

Definition 4.8 a and b are **incomparable** if neither $a \preceq b$ nor $b \preceq a$.

Definition 4.9 a and b are **comparable** if $a \preceq b$ or $b \preceq a$.

E.g., for subsets of $\mathbb{N}$, $\{1, 2, 3\}$ and $\{2, 3, 4\}$ are incomparable.
However, a partial order need not have incomparable elements.
As a special case, we can have a partial order in which there is a specified order between every pair of elements.
Such partial orders are called "total orders."

Definition 4.10 A poset $(S, \preceq)$ is **totally ordered** if $(\forall a, b \in S)[a \preceq b \vee b \preceq a]$.

Examples:

- $S = \mathbb{N}, \preceq = \leq$
- $S = \mathbb{N}, \preceq = |$, NOT because neither $3|5$ nor $5|3$
- $S = P(\mathbb{N}), \preceq = \subseteq$, NOT because neither $\{3\} \subseteq \{5\}$ nor $\{5\} \subseteq \{3\}$
- Lexicographic order on pairs of natural numbers, defined by: $(a_1, a_2) \preceq (b_1, b_2)$ if and only if either $a_1 < b_1$, or else $a_1 = b_1$ and $a_2 \leq b_2$.
  Do some examples here. $(2, 37) \preceq (2, 38) \preceq (3, 2)$.
  "Lexicographic" because like dictionary.

The Hasse diagram of a total order looks like a line.

This has been the basic material.
Book has a bit more:
minimal, maximal elements (nothing smaller, nothing larger)
lower bounds, upper bounds (for a subset – $\leq$ everything in the subset, or $\geq$)
lub, glb,
lattices.
Read these parts.

4.2 Topological Sorting

Sometimes when we have a partial order, e.g., of tasks to be performed, we want to obtain a consistent total order.
That is, an order in which to perform all the tasks, one at a time, so as not to conflict with the precedence requirements.

The task of finding an ordering that is consistent with a partial order is known as "topological sorting". Maybe because the sort is based only on the shape (topology) of the poset, and not on the actual values.

Definition 4.11 A **topological sort** of a finite poset $(A, \preceq)$ is a total ordering of all the elements of A, $a_1, a_2, \cdots, a_n$
in such a way that $\forall i < j$, either $a_i \prec a_j$ or a_i and a_j are incomparable in $(A, \preceq)$ (equivalently, it is not the case that $a_j \prec a_i$).

For example, underwear, shirt, Lsock, Rsock, pants, sweater, Lshoe, Rshoe, belt, jacket is a topological sort of how to dress.

One of the nice facts about posets is that such an ordering always exists and is even easy to find:

Theorem 4.12 Every finite poset has a topological sort.

We'll prove this by induction.
On the number of elements in the domain of the poset.
The basic idea is to pick a "smallest" element to start and then proceed recursively.

Definition 4.13 A **minimal** element a in a poset $(A, \preceq)$ is one for which $(\forall b \in A)[a \preceq b \vee a \text{ and } b \text{ are incomparable }]$. Equivalently, it is an element a for which no $b \prec a$.

Notice that there can be more than one minimal element. There are 4 in the clothes example: Lsock, Rsock, underwear, shirt.

Lemma 4.14 Every finite poset $(A, \preceq)$ has a minimal element.

Proof. By induction on the number of elements in the poset.
The base case is clear. Suppose it is true for all size-n posets.
Consider any size $n + 1$ poset. Delete some element a from the poset.
This leaves a poset on n elements (you can verify this).
So the smaller poset has a minimal element m.

Now consider two cases. First suppose $a \not\preceq m$.
The m is a minimal element of the whole poset.
Now suppose $a \preceq m$. Then I claim a is a minimal element of the whole poset.

To prove this claim we argue by contradiction.
If a is not minimal then some $b \preceq a$.
But then, by transitivity, since $a \preceq m$, we have $b \preceq m$.
But this would prevent m from being minimal in the poset with a removed.
This contradicts our finding of m. ∎

Note that an infinite poset might have no minimal element. Consider $\mathbb{Z}$.

Now we can prove the existence of the topological sort.

Proof. By (ordinary) induction.
$P(n)$ = any poset with n elements has a topological sort.
Base case ($P(1)$): for a poset with one element the one element makes a topological sort.

Inductive step: Assume ($P(n)$). Consider a poset of $n+1$ elements.
By lemma it has a minimal element u.
Consider the same relation on $A - \{u\}$.
It is easy to check that it is still reflexive, transitive, and anti-symmetric.
So by the inductive hypothesis, $A - \{u\}$ has a topological sort.
We put u first and now have a topological sort on the whole thing.
(According to the definition of topological sort – u can't be in the wrong order w.r.t. any of the later elements.)

Thus every finite poset has a topological sort, by induction. ∎

E.g.: Dressing example:
Construct ordering by picking one at a time.
In each case, look at the remaining partial order for remaining elements.
Lsock, shirt, sweater, Rsock, underwear, pants, Lshoe, belt, jacket, Rshoe

I'm not sure if infinite posets always have topological sorts—I think this might be equivalent to the well ordering property, which is equivalent to the axiom of choice, a hotly debated mathematical axiom.