Relations

A "relation" is a fundamental mathematical notion expressing a relationship between elements of sets.

Definition 0.1. A *binary relation from a set A to a set B* is a subset $R \subseteq A \times B$.

So, R is a set of ordered pairs. We often write $a \sim_R b$ or aRb to mean that $(a, b) \in R$.

Functions, for example, are a type of relation. The abstract notion of a relation is useful both in mathematics and in practice for modeling many different sorts of relationships. It's the basis of the *relational database* model, the standard data model for practical data processing systems.

Many times we will talk about a "relation on the set A", which means that the relation is a subset of $A \times A$. 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. In this class, we will focus only on binary relations. Here are some examples:

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

1 Properties of Relations

Once we have modeled something abstractly as a relation, we can talk about its properties without referring to the original problem domain. For a relation on a set A there are several standard properties of relations that occur commonly. Later on we will use these properties to classify different types of relations.

Definition 1.1. A binary relation R on A is:

1. *reflexive* if for every $a \in A$, $a \sim_R a$.
2. *symmetric* if for every $a, b \in A$, $a \sim_R b$ implies $b \sim_R a$.
3. *antisymmetric* if for every $a, b \in A$, $a \sim_R b$ and $b \sim_R a$ implies $a = b$.
4. *asymmetric* if for every $a, b \in A$, $a \sim_R b$ implies $\neg(b \sim_R a)$.
5. *transitive* if for every $a, b, c \in A$, $a \sim_R b$ and $b \sim_R c$ implies $a \sim_R c$.

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 relations from Example :

- Relation 3 is reflexive, symmetric, transitive.
- Relation 4 is reflexive, transitive. Not necessarily symmetric, since roads could be one-way (consider Boston), but in actuality . . . . But definitely not antisymmetric.
- Relation 5 is symmetric but not transitive. Whether it is reflexive is open to interpretation.
- Relation 6 likewise.
- Relation 7 is (unfortunately) not symmetric. Not antisymmetric. Not transitive. Not even reflexive!
- Relation 8 is reflexive, antisymmetric, transitive.
- Relation 9 is not reflexive. It is symmetric. It is not transitive. {even naturals}∩{odd naturals} is finite (empty), but not {even naturals} ∩ {even naturals}.
- Relation 10 is only symmetric.
- Relation 11 is reflexive, antisymmetric and transitive.

2 Representation

There are many different ways of representing relations. One way is to describe them by properties, as we did above. For infinite sets, that's about all we can do. But for finite sets, we usually use some method that explicitly enumerates all the elements of the relation. Some alternatives are lists, matrices and graphs. Why do we have so many different ways to represent relations? Different representations may be more efficient for encoding different problems and also tend to highlight different properties of the relation.

2.0.1 Lists

A finite relation from set A to set B can be represented by a list of all the pairs.

Example 2.1. The relation from $\{0, 1, 2, 3\}$ to $\{a, b, c\}$ defined by the list:

$\{(0, a), (0, c), (1, c), (2, b), (1, a)\}$.

Example 2.2. The divisibility relation on natural numbers $\{1, \ldots, 12\}$ is represented by the list:

$$\{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(1,7),(1,8),(1,9),(1,10),(1,11),(1,12),(2,2),(2,4),\\ (2,6),(2,8),(2,10),(2,12),(3,3),(3,6),(3,9),(3,12),(4,4),(4,8),(4,12),(5,5),(6,6),\\ (6,12),(7,7),(8,8),(9,9),(10,10),(11,11),(12,12)\}.$$

We can recognize certain properties by examining this representation:

Reflexivity: Contains all pairs (a, a).

Symmetry: Contains (a, b) then contains (b, a).

Transitivity: Contains (a, b) and (b, c) then contains (a, c).

2.0.2 Boolean Matrices

Boolean matrices are a convenient representation for representing relations in computer programs. The rows are for elements of A, columns for B, and for every entry there is a 1 if the pair is in the relation, 0 otherwise.

Example 2.3. The relation from Example 2.1 is represented by the matrix

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

Example 2.4. The divisibility relation over $\{1, 2, \ldots, 12\}$ is represented by the enormous 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

Again, properties can by recognized by examining the representation:

Reflexivity the major diagonal is all 1.

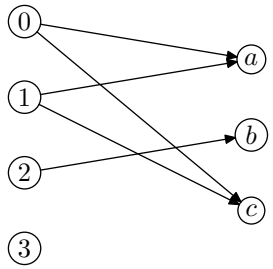
Symmetry: the matrix is clearly not symmetric across the major diagonal.

Transitivity: not so obvious . . .

2.0.3 Digraphs

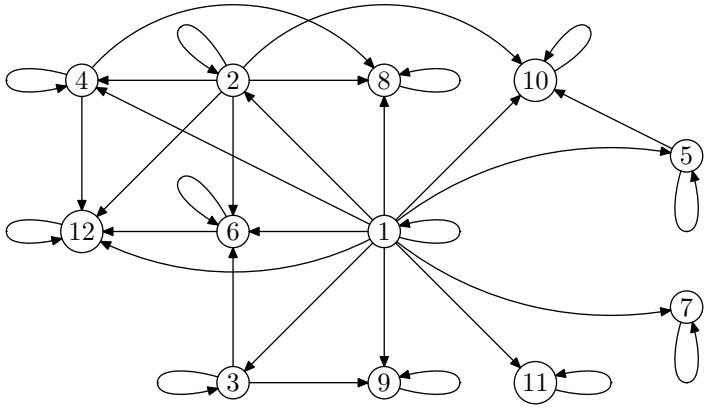
We can draw a picture of a relation $R \subseteq A \times B$ by drawing a dot for every element of A, a dot for every element of B, and an arrow from $a \in A$ to $b \in B$ iff aRb. Such a picture is called a *directed graph*, or *digraph* for short.

Example 2.5. The relation from Example 2.1 is represented by the digraph



Digraphs are mainly used for relations where $A = B$, i.e., for relations on a finite set A. To represent such 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.

Example 2.6. 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.

3 Operations on Relations

3.1 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."

Example 3.1. Inverse of "is taking class" (Relation 1 in Example) is the relation "has as a student" on the set {classes at MIT} $\times$ {students at MIT}; a relation from classes to students.

We can translate the inverse operation on relation to operations on the various representations of a relation. Given the matrix for R, we can get the matrix for R^{-1} by transposing the matrix for R (note that the inverse of a relation is not the same thing as the inverse of the matrix representation). Given a digraph for R, we get the graph for R^{-1} by reversing every edge in the original digraph.

3.2 Composition

The *composition* of relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ is the relation

$$R_2 \circ R_1 = \{(a, c) \mid (\exists b)((a, b) \in R_1) \wedge ((b, c) \in R_2)\}.$$

In words, the pair (a, c) is in $R_2 \circ R_1$ if there exists an element b such that the pair (a, b) is in R_1 and the pair (b, c) is in R_2. Another way of thinking about this is that a "path" exists from element a to c via some element in the set B[1].

[1]Notice that $R_2 \circ R_1$ and $R_1 \circ R_2$ are different. The symbol $\circ$ is a source of eternal confusion in mathematics—if you read a book you should always check how the authors define $\circ$—some of them define composition the other way around.

Example 3.2. The composition of the relation "is taking class" (Relation 1 in Example) with the relation "has lecture in" (Relation 2 in Example) is the relation "should go to lecture in", a relation from {students at MIT} to {rooms at MIT}.

Example 3.3. Composition of the parent-of relation with itself gives grandparent-of. Composition of the child-of relation with the parent-of relation gives the sibling-of relation. (Here we relax the meaning of sibling to include that a person is the sibling of him/herself.) Does composition of parent-of with child-of give married-to/domestic partners? No, because that misses childless couples.

Example 3.4. Let B be the set of boys, G be the set of girls, $R_1 \subseteq B \times G$ consist of all pairs (b, g) such that b is madly in love with g, and $R_2 \subseteq G \times B$ consist of all pairs (g, b) such that g is madly in love with b. What are the relations $R_2 \circ R_1$ and $R_1 \circ R_2$, respectively?

3.2.1 Computing Composition and Path Lengths

If we represent the relations as matrices, then we can compute the composition by a form of "boolean" matrix multiplication, where $+$ is replaced by $\vee$ (Boolean OR) and $\times$ is replaced by $\wedge$ (Boolean AND).

Example 3.5. Let R_1 be the relation from Example 2.1:

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

Let R_2 be the 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

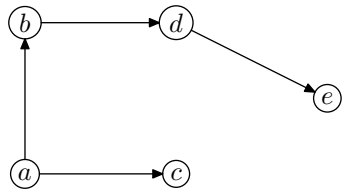
Then $R_2 \circ R_1 = \{(0, d), (0, e), (0, f), (1, d), (1, e), (1, f), (2, e)\}$, that is,

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

A relation on a set A can be composed with itself. The composition $R \circ R$ of R with itself is written R^2. Similarly R^n denotes R composed with itself n times. R^n can be recursively defined: $R^1 = R$, $R^n = R \circ R^{n-1}$.

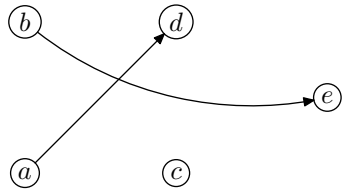
Example 3.6. Consider the relation $R = \{(a,b),(a,c),(b,d),(d,e)\}$ or:

	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



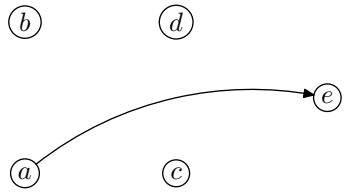
R^2 will be:

	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



R^3 will be:

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



Definition 3.7. A *path* in a relation R is a sequence $a_0, \ldots, a_k$ 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 digraph 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). A *simple path* is a path with no repeated vertices.

Lemma 3.8. $R^n = \{(a,b) \mid$ *there is a length* n *path from* a *to* b *in* $R\}$

Proof. By induction. Let

$$P(n) ::= R^n = \{(a,b) \mid \text{there is a length } n \text{ path from } a \text{ to } b \text{ in } R\}.$$

The base case is clear. There is exactly one edge from a to b for every $(a,b) \in R$. And there is exactly one pair $(a,b) \in R$ for every edge from a to b, $P(1)$ is true. Note that since the induction hypothesis is an equality we have to prove both sides.

For the inductive step, suppose $P(n)$ is true.

First consider a path $a_0, \ldots, a_{n+1}$ in R. This is a path $a_0, \ldots, a_n$ in R followed by a pair (a_n, a_{n+1}) of R. By the inductive hypothesis, we can assume that $(a_0, a_n) \in R^n$. And we have already

mentioned that $(a_n, a_{n+1}) \in R$. Therefore $(a_0, a_{n+1}) \in R^{n+1}$ by the definition of composition. Thus, every path of length $n+1$ corresponds to a relation in R^{n+1}.

Now consider a pair $(a, b) \in R^{n+1}$. By the definition of composition, there exists a c such that $(a, c) \in R^n$ and $(c, b) \in R$. By the inductive hypothesis, we can assume that (a, c) corresponds to a length n path from a to c, and since $(c, b) \in R$ there is an edge from c to b. Thus, there is a length $n+1$ path from a to b. To conclude, $P(n) \longrightarrow P(n+1)$. □

3.3 Closure

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

Definition 3.9. 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).

As a general principle, there are two ways to construct a closure of R with respect to property P: We can either start with R and add as few pairs as possible until the new relation has property P; or we can start with the largest possible relation (which is $A \times A$ for a relation on A) and then remove as many not-in-R pairs as possible while preserving the property P.

3.3.1 The Reflexive Closure

Lemma 3.10. *Let R be a relation on the set A. The reflexive closure of R is $S = R \cup \{(a, a), \forall a \in A\}$.*

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. □

Example 3.11. Let $R = \{(a, b)(a, c)(b, d)(d, e)\}$, then the reflexive closure of R is

$$\{(a, b)(a, c)(b, d)(d, e)(a, a)(b, b)(c, c)(d, d)(e, e)\}.$$

3.3.2 The Symmetric Closure

Lemma 3.12. *Let R be a relation on the set A. 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 $S = R \cup R^{-1} \subseteq T$. □

Example 3.13. Let $R = \{(a, b)(a, c)(b, d)(d, e)\}$, then the symmetric closure of R is

$$\{(a, b)(a, c)(b, d)(d, e)(b, a)(c, a)(d, b)(e, d)\}$$

3.3.3 The transitive closure

The transitive closure is a bit more complicated than the closures above.

Lemma 3.14. *Let R be a relation on the set A. The transitive closure of a relation R is the set*

$$S = \{(a, b) \in A^2 \textit{given there is a path from } a \textit{ to } b \textit{ in } R\}.$$

Proof. Obviously, $R \subseteq S$. Next, we show that S is transitive. Suppose $(a, b) \in S$ and $(b, c) \in S$. 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.

Finally, we need to show that S is the smallest transitive relation containing R. So consider any transitive relation T containing R. We have to show that $S \subseteq T$. Assume for contradiction that $S \nsubseteq T$. 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 R where $k \geq 1$ and $(a_0, a_k) \notin T$. Call this a *missing path*. Now let M be the set of missing paths.

Now we use well-ordering. We have just claimed that the set M of missing paths 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.

Case 1: $m = 1$. Then 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.

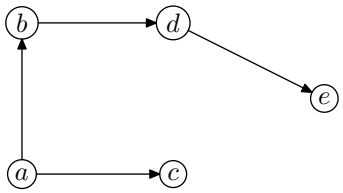
Case 2: $m > 1$. Then $s_1, \ldots, s_{m-1}$ is a path in R ($m - 1 > 0$). But it is shorter than our original shortest missing path, so cannot be missing. Thus $(s_1, s_{m-1}) \in T$. Also we have $(s_{m-1}, s_m) \in T$ since $R \subseteq T$. Thus by transitivity of T, $(s_1, s_m) \in T$, a contradiction.

We get a contradiction either way, so our assumption (that $S \nsubseteq T$) is false. This completes the proof. □

Wait a minute. Well-ordering is applied to sets of *numbers*; we applied it to a set of paths! 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.

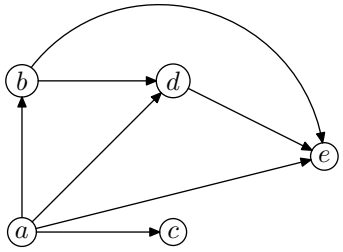
Example 3.15. The transitive closure of the relation

	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



is

	a	b	c	d	e
a	0	1	1	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



3.3.4 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 $R^n = \{(a, b) given$there is a length n path from a to b in $R\}$. This means we can write the transitive closure of R as $R \cup R^2 \cup R^3 \cup \cdots$. Better—since we know how to do composition—but still a problem: there are infinitely many terms!

Lemma 3.16. *Suppose A has n elements and that R is a relation on A. Let a and b be elements of A and suppose that there is a path from a to b in R. Then, there is a path of length at most n from a to b in R.*

Proof. We'll use well-ordering (again). Consider the shortest path $a = a_0, a_1, \ldots, a_k$ 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 at most n, namely $R^1 \cup R^2 \cup \cdots \cup R^n$.

4 Equivalence Relations and Partitions

We can use properties of relations to classify them into different types. We will be considering two important types of relations, *equivalence relations* and *partial orders*.

Definition 4.1. An *equivalence relation* is a relation that is reflexive, symmetric and transitive.

For example, the "roommates" relation is an equivalence relation. So is "same size as", and "on same Ethernet hub". A trivial example is the $=$ relation on natural numbers. The hallmark of equivalence relation is the word *same*. It provides a way to hide unimportant differences. Using an equivalence relation we can actually partition the universe into subsets of things that are the "same", in a natural way.

Definition 4.2. A *partition* of a set A is a collection of subsets $\{A_1, \ldots, A_k\}$ 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.

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

Lemma 4.3. *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. Consider some arbitrary $a, b \in A$. If either $[a] = [b]$ or $[a] \cap [b] = \emptyset$ 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.) 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 $d \sim_R c$. □

Note that all three properties of equivalence relations were used in this proof. Checking that the proof uses all available assumptions if usually a good sanity check when writing proofs—if one of the properties you assumed were not needed, you have either made a mistake or proven a much more strong theorem than you thought—*e.g.*, if you didn't use transitivity anywhere in the proof of Lemma 4.3, you would be proving that any reflexive symmetric relation produces a partition, which is false.

Lemma 4.4. *Any partition $\{A_1, \ldots, A_k\}$ of A defines an equivalence relation by letting $a \sim_R b$ iff a and b are in the same A_i.*

Proof. 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, *i.e.*, $a \sim_R a$.

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.

4.1 Integers modulo m

A familiar and important equivalence relation on integers (positive, negative and 0) is:

Definition 4.5. If a and b are integers, then we say that $a \equiv b \pmod{m}$ if $m \mid (a - b)$.

$a \equiv b \pmod{m}$ is pronounced "a is equivalent to b modulo m". An equivalent formulation says that $a = b + km$ for some integer k.

Theorem 4.6. *Equality modulo m is an equivalence relation.*

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

Reflexive: Clearly $m \mid (a - a) = 0$, so $a \equiv a \pmod{m}$.

Symmetric: If $a = b + km$ then $b = a + (-k)m$.

Transitive: Suppose $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$. 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 the set $[a] = \{b \in \mathbb{Z} \mid a \equiv b \pmod{m}\}$, 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 representatives 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 4.7. *If $a \equiv x \pmod{m}$ and $b \equiv y \pmod{m}$ then $(a + b) \equiv (x + y) \pmod{m}$.*

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—then 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.

5 Partial Orders

Partial orders are another type of binary relation that is very important in computer science. They have applications to task scheduling, database concurrency control, and logical time in distributed computing,

Definition 5.1. A binary relation $R \subseteq A \times A$ is a *partial order* if it is reflexive, transitive, and antisymmetric.

Recall that antisymmetric mean $aRb \wedge bRa \Rightarrow a = b$, or $\forall a \neq b, aRb \Rightarrow \neg bRa$. In other words this relation is *never* symmetric! This single property is what distinguishes it from an equivalence relation. The reflexivity, antisymmetry and transitivity properties are abstract properties that generally describe "ordering" relationships.

For a partial order relation 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 it is 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":

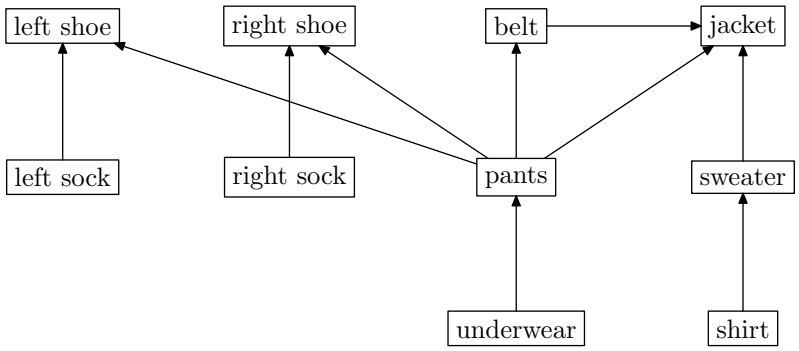
Definition 5.2. A set A together with a partial order $\preceq$ is called a *poset* $(A, \preceq)$.

Example 5.3. Consider the following relations:

- $A = \mathbb{N}, R = \leq$, easy to check reflexive, transitive, antisymmetric
- $A = \mathbb{N}, R = \geq$, same.
- $A = \mathbb{N}, R = <$, *not* because not reflexive
- $A = \mathbb{N}, R = |$ (divides), easy to check reflexive, transitive, antisymmetric
- $A = \mathcal{P}(\mathbb{N}), R = \subseteq$, 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 partial order because it is not true that $aRb \wedge bRa \Rightarrow a = b$. In fact, it is symmetric and transitive. Equivalence relation.
- $A =$ "set of all propositions", $R = \Longrightarrow$, **not** because it's not antisymmetric. Not symmetric either, so not equivalence relation.

5.1 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 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 5.4. A *cycle* is a path that ends where it started (*i.e.*, the last vertex equals the first).

Definition 5.5. A *directed acyclic graph (DAG)* is a directed graph with no cycles.

Lemma 5.6. *Any partial order is a DAG.*

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

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."

Lemma 5.7. *The transitive reflexive closure of a DAG is a partial order.*

Proof. Let the DAG be R and its transitive reflexive closure S. S is transitive and reflexive; we just need to prove that it is antisymmetric. We do so by contradiction. Suppose that there exists some a, b such that $a \neq b$, $a \sim_S b$, and $b \sim_S a$. In other words, there is a path from a to b and a path from b to a in R. If we attach these two paths, we get a path from a to a in R, *i.e.*, R contains a cycle. This contradicts the assumption that R is acyclic. □

5.2 Partial vs. Total Orders

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

Example 5.8. Lshoe and Rshoe have no prescribed ordering between them.

Example 5.9. 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 5.10. We say that a and b are *incomparable* if neither $a \preceq b$ nor $b \preceq a$, and that they are *comparable* if $a \preceq b$ or $b \preceq a$.

Example 5.11. 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.

Definition 5.12. A poset $(S, \preceq)$ is *totally ordered* if $(\forall a, b \in S)[a \preceq b \vee b \preceq a]$.

The DAG for a total order looks like a line.

5.3 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*—probably because the sort is based only on the shape, *i.e.*, topology, of the poset, and not on the actual values.

Definition 5.13. 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 for all $i < j$, either $a_i \preceq a_j$ or a_i and a_j are incomparable.

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 5.14. *Every finite poset has a topological sort.*

The basic idea to prove this theorem is to pick off a "first" element and then proceed inductively.

Definition 5.15. A *minimal* element a in a poset $(A, \preceq)$ is one for which $(\forall b \in A)[a \preceq b \vee a$ and b are incomparable$]$. Equivalently, it is an element a for which $(\nexists b \in A)[b \prec a]$.

Lemma 5.16. *Every finite poset $(A, \preceq)$ has a minimal element.*

Proof. For every element $a \in A$, let $p_a = \{b \in A \text{ given } b \prec a\}$, *i.e.*, p_a is the set of predecessors of a according to the partial order. It is enough to show that there exists an $a \in A$ such that $p_a = \emptyset$; to accomplish this, we use well-ordering.

Let $P = \{p_a \text{ given } a \in A\}$. Now let a' be an element corresponding to a set $p_{a'}$ of minimum cardinality. We now that such an a' exists by applying the well-ordering principle to the subset $\{|p| : p \in P\}$ of $\mathbb{N}$. There can be several sets in P of minimum cardinality, but that doesn't matter—we pick $p_{a'}$ to be one of the sets. We now prove by contradiction that $p_{a'} = \emptyset$.

Suppose that $|p_{a'}| > 0$, *i.e.*, that $p_{a'} \neq \emptyset$. Then there exists some $b' \in A$ such that $b' \prec a'$. Consider the set $p_{b'}$. We now claim that $p_{b'} \subset p_{a'}$. Since $c \in p_{b'}$ implies that $c \prec b'$, we obtain, by transitivity, that $c \in p_{b'}$ implies that $c \prec a'$, *i.e.*, every element in $p_{b'}$ is also an element of $p_{a'}$, or, equivalently, $p_{b'} \subseteq p_{a'}$. Furthermore, $b' \in p_{a'}$ since $b' \prec a'$, but $b' \notin p_{b'}$ since $b' \not\prec b'$. Thus $p_{b'} \subset p_{a'}$. But this contradicts our assumption that $p_{a'}$ is a set of minimum cardinality. Thus, $p_{a'} = \emptyset$. □

Example 5.17. Consider the dressing example. Construct an ordering by picking one item at a time. At each step, look at the poset formed by the remaining elements. Lsock, shirt, sweater, Rsock, underwear, pants, Lshoe, belt, jacket, Rshoe

Example 5.18. Subsets of $\{1, 2, 3, 4\}$:

$\emptyset$ is the unique minimal element, then we have choices, *e.g.*, do: $\{1\}$, $\{2\}$, $\{1, 2\}$, $\{3\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$, $\{4\}$, $\{1, 4\}$, $\{2, 4\}$, $\{3, 4\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, $\{2, 3, 4\}$, $\{1, 2, 3, 4\}$

5.4 Parallel Task Scheduling

When elements of a poset are tasks that need to be done and the partial order is precedence constraints, topological sorting provides us with a legal way to execute tasks sequentially, *i.e.*, without violating any precedence constraints. But what if we have the ability to execute more than one task at the same time? For example, say tasks are programs, partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one. How should we schedule the tasks? Our goal should be to minimize the total *time* to complete all the tasks. For simplicity, let's say all the tasks take the same amount of time and all the processors are identical.

So, given a finite poset of tasks, how long does it take to do them all, in an optimal parallel schedule? We can use partial order concepts to analyze this problem.

On the clothes example, we could do all the minimal elements first (Lsock, Rsock, underwear, shirt), remove them and repeat. We'd need lots of hands, or maybe dressing servants. We can do pants and sweater next, and then Lshoe, Rshoe, and belt, and finally jacket.

We can't do any better, because the sequence underwear, pants, belt, jacket must be done in that order. A sequence like this is called a chain.

Definition 5.19. A *chain* in a poset is a sequence of elements of the domain, each of which is smaller than the next in the partial order ($\preceq$ and $\neq$). The *length* of a chain is the number of elements in the chain.

Note that a chain is just a path in the corresponding graph.

Clearly, the parallel time is at least length of any chain. For if we used less time, then two tasks in the chain would have to be done at the same time. (This is "obvious," but is formalized as an application of the "pigeonhole principle" we will study shortly.) But by definition of chains this violates precedence constraints. A longest chain is also known as a *critical path*. So we need at least t steps, where t is the length of the longest chain. Fortunately, it is always possible to use only t:

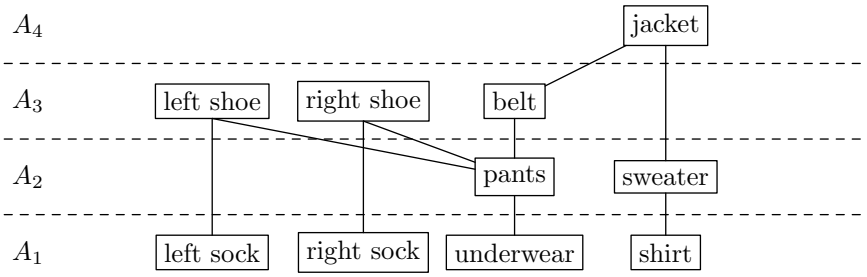
Theorem 5.20. *Given any finite poset $(A, \preceq)$ for which the longest chain has length t, it is possible to partition A into t subsets, $A_1, A_2, \cdots, A_t$ such that*

$$(\forall i \in \{1, 2, \ldots, t\})(\forall a \in A_i)(\forall b \prec a)\big[b \in A_1 \cup A_2 \cup \cdots \cup A_{i-1}\big].$$

That is, we can divide up the tasks into t groups so that for each group A_i, all tasks that have to precede tasks in A_i are in smaller-numbered groups.

Corollary 5.21. *For $(A, \preceq)$ and t as above, it is possible to schedule all tasks in t steps.*

Proof. For all i, schedule all elements of A_i at time i. This satisfies the precedence requirements, because all tasks that must precede a task are scheduled at preceding times. □



Corollary 5.22. *parallel time = length of longest chain*

So it remains to prove the partition theorem:

Proof of Theorem 5.20. Construct the sets A_i as follows:

$$A_i = \{a \in A \text{ given longest chain ending in } a \text{ has length } i\}.$$

This gives just t sets, because the longest chain has length t. Also, each $a \in A$ belongs to exactly one A_i. To complete the proof, we also need to show

$$(\forall i \in \{1, 2, \ldots, t\})(\forall a \in A_i)(\forall b \prec a)\big[b \in A_1 \cup A_2 \cup \cdots \cup A_{i-1}\big].$$

The proof is by contradiction. Assume that $a \in A_i$ and that there exists a $b \notin A_1 \cup A_2 \cup \cdots \cup A_{i-1}$ such that $b \prec a$. Then there is a chain of length exceeding $i - 1$ ending in b. This means, since $b \prec a$, that there is a chain of length $> i$ ending in a, which means that $a \notin A_i$. □

So with an unlimited number of processors, the time to complete all the tasks is the length of the longest chain. It turns out that this theorem is good for more than parallel scheduling. It is usually stated as follows.

Definition 5.23. An *antichain* is a set of incomparable elements.

Corollary 5.24. *If t is the length of the longest chain in a poset $(A, \preceq)$ then A can be partitioned into t antichains.*

Proof. Let the antichains be the sets A_i defined as in the proof of Theorem 5.20. We now claim that the elements in those sets are incomparable. Suppose that there exists $a, b \in A_i$ such that $a \neq b$ and a and b are comparable. Then either $a \prec b$ or $b \prec a$, which—again by the proof of Theorem 5.20—contradicts the assumption that a and b are in the same A_i. □

5.4.1 Dilworth's Theorem

We can use the above corollary to prove a famous result about posets:

Theorem 5.25 (Dilworth). *For all t, every poset with n elements must have either a chain of size greater than t or an antichain of size at least n/t.*

Proof. Assume there is no chain of length greater than t. So, longest chain has length at most t. Then by Corollary 5.24, the n elements can be partitioned into at most t antichains. Let ℓ be the size of the largest antichain. Since there are at most t antichains, every antichain contains at most ℓ elements, and an element belongs to exactly one antichain, $\ell t \geq n$. So there is an antichain with at least n/t elements. □

Corollary 5.26. *Every poset with n elements has a chain of length greater than $\sqrt{n}$ or an antichain of size at least $\sqrt{n}$.*

Proof. Set $t = \sqrt{n}$ in Theorem 5.25. □

Example 5.27. In the dressing poset, $n = 10$. Try $t = 3$. Has a chain of length 4. Try $t = 4$. Has no chain of length 5, but has an antichain of size $4 \geq 10/4$.

Posets arise in all sorts of contexts, and when they do, Dilworth's theorem can have interesting implications.

5.4.2 Increasing and Decreasing Sequences

Theorem 5.28. *In any sequence of n different numbers, there is either an increasing subsequence of length greater than $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.*

Example 5.29. $\langle 6, 4, 7, 9, 1, 2, 5, 3, 8 \rangle$ has the decreasing sequence $\langle 6, 4, 1 \rangle$ and the increasing sequence $\langle 1, 2, 3, 8 \rangle$.

We can prove this using Dilworth's theorem; the trick is to define the appropriate poset. The domain is the set of values in the sequence. For the ordering, define $a \preceq b$ if either $a = b$, or else ($a < b$ and a comes before b in the sequence). You should check that this is reflexive (stated explicitly), transitive, and antisymmetric. A chain corresponds to a sequence that increases in value and moves to the right, that is, an increasing sequence. But what does an antichain correspond to?

Lemma 5.30. *If a and b are incomparable (under the partial order) and $a > b$ (as numbers) then a precedes b in the sequence.*

Proof. By contradiction. Assume b precedes a in the sequence. Then $b < a$ and b precedes a in the sequence, so $b \preceq a$ in the partial order. This contradicts our assumption that a and b are incomparable. □

We extend this lemma to more than 2 elements:

Lemma 5.31. *If $a_1, a_2, \cdots, a_t$ are incomparable and $a_1 > a_2 > \cdots > a_t$ then $a_1, a_2, \cdots a_t$ form a decreasing subsequence.*

Proof. For all i, the fact that $a_i > a_{i+1}$ implies that a_i precedes a_{i+1} in the sequence, by the previous lemma. □

So given an antichain, arrange the elements so that they are in decreasing order and the "left of" relation follows, giving a decreasing subsequence. Dilworth's theorem implies that there is either a chain of size greater than $\sqrt{n}$ or an antichain of size at most $\sqrt{n}$. By the analysis above, this yields either an increasing sequence of length greater than $\sqrt{n}$ or a decreasing subsequence of length at most $\sqrt{n}$.