

# Relations

## 1 Relations

A “relation” is a mathematical tool used to describe relationships between set elements. Relations are widely used in computer science, especially in databases and scheduling applications.

Formally, a **relation** from a set  $A$  to a set  $B$  is a subset  $R \subseteq A \times B$ . For example, suppose that  $A$  is a set of students, and  $B$  is a set of classes. Then we might consider the relation  $R$  consisting of all pairs  $(a, b)$  such that student  $a$  is taking class  $b$ :

$$R = \{(a, b) \mid \text{student } a \text{ is taking class } b\}$$

Thus, student  $a$  is taking class  $b$  if and only if  $(a, b) \in R$ . There are a couple common, alternative ways of writing  $(a, b) \in R$  when we’re working with relations:  $aRb$  and  $a \sim_R b$ . The motivation for these alternative notations will become clear shortly.

### 1.1 Relations on One Set

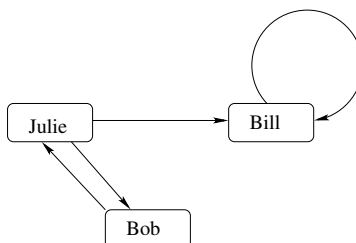
We’re mainly going to focus on relationships between elements of a single set; that is, relations from a set  $A$  to a set  $B$  where  $A = B$ . Thus, a **relation** on a set  $A$  is a subset  $R \subseteq A \times A$ . Here are some examples:

- Let  $A$  be a set of people and the relation  $R$  describe who likes whom; that is,  $(x, y) \in R$  if and only if  $x$  likes  $y$ .
- Let  $A$  be cities. Then we can define a relation  $R$  such that  $xRy$  if and only if there is a nonstop flight from city  $x$  to city  $y$ .
- Let  $A = \mathbb{Z}$ , and let  $xRy$  hold if and only if  $x \equiv y \pmod{5}$ .
- Let  $A = \mathbb{N}$ , and let  $xRy$  hold if and only if  $x \mid y$ .
- Let  $A = \mathbb{N}$ , and let  $xRy$  hold if and only if  $x \leq y$ .

The last examples clarify the reason for using  $xRy$  or  $x \sim_R y$  to indicate that the relation  $R$  holds between  $x$  and  $y$ : many common relations ( $<$ ,  $\leq$ ,  $=$ ,  $\mid$ ,  $\equiv$ ) are expressed with the relational symbol in the middle.

## 1.2 Relations and Directed Graphs

A relation can be modeled very nicely with a directed graph. For example, the directed graph below describes the “likes” relation on a set of three people:



From this directed graph, we can conclude that:

- Julie likes Bill and Bob, but not herself.
- Bill likes only himself.
- Bob likes Julie, but not himself.

In fact, everything about the “likes” relation is conveyed by this graph, and nothing more. This is no coincidence; a set  $A$  together with a relation  $R$  is precisely the same thing as a directed graph  $G = (V, E)$  with vertex set  $V = A$  and edge set  $E = R$ .

From this example, you may suppose that this is going to be another one of those racy 6.042 lectures designed to get your attention. Well, this is not your lucky day. There will be no sex mentioned for the rest of the lecture. Instead, we will be discussing how to use the ideas behind relations to help you put your clothes *on*, rather than off.

## 2 Properties of Relations

Many relations that arise in practice possess some standard properties. A relation  $R$  on set  $A$  is:

1. **reflexive** if  $xRx$  for all  $x$  in  $A$ .  
(Everyone likes themselves.)
2. **symmetric** if for all  $x, y \in A$ ,  $xRy$  implies  $yRx$ .  
(If  $x$  likes  $y$ , then  $y$  likes  $x$ .)
3. **antisymmetric** if for all  $x, y \in A$ ,  $xRy$  and  $yRx$  imply that  $x = y$ .  
(If  $x$  likes  $y$  and  $y$  likes  $x$ , then  $x$  and  $y$  are the same person.)

4. **transitive** if for all  $x, y, z \in A$ ,  $xRy$  and  $yRz$  imply  $xRz$ .

(If  $x$  likes  $y$  and  $y$  likes  $z$ , then  $x$  also likes  $z$ .)

Let's see which of these properties hold for some of the relations we've considered so far:

	reflexive?	symmetric?	antisymmetric?	transitive?
$x \equiv y \pmod{5}$	yes	yes	no	yes
$x \mid y$	yes	no	yes	yes
$x \leq y$	yes	no	yes	yes

The two different yes/not patterns in this table are both extremely common. A relation with the first pattern of properties (like  $\equiv$ ) is called an “equivalence relation”, and a relation with the second pattern (like  $\mid$  and  $\leq$ ) is called a “partially-ordered set”. The rest of this lecture focuses on just these two types of relation.

### 3 Equivalence Relations

A relation is an **equivalence relation** if it is reflexive, symmetric, and transitive. Congruence modulo  $n$  is an excellent example of an equivalence relation:

- It is reflexive because  $x \equiv x \pmod{n}$ .
- It is symmetric because  $x \equiv y \pmod{n}$  implies  $y \equiv x \pmod{n}$ .
- It is transitive because  $x \equiv y \pmod{n}$  and  $y \equiv z \pmod{n}$  imply that  $x \equiv z \pmod{n}$ .

There is an even more well-known example of an equivalence relation: equality itself. Thus, an equivalence relation is a relation that shares some key properties with  $=$ .

#### 3.1 Partitions

There is another way to think about equivalence relations, but we'll need a couple definitions to understand this alternative perspective.

Suppose that  $R$  is an equivalence relation on a set  $A$ . Then the **equivalence class** of an element  $x \in A$  is the set of all elements in  $A$  related to  $x$  by  $R$ . The equivalence class of  $x$  is denoted  $[x]$ . Thus, in symbols:

$$[x] = \{y \mid xRy\}$$

For example, suppose that  $A = \mathbb{Z}$  and  $xRy$  means that  $x \equiv y \pmod{5}$ . Then:

$$[7] = \{\dots, -3, 2, 7, 12, 17, 22, \dots\}$$

Notice that 7, 12, 17, etc. all have the same equivalence class; that is,  $[7] = [12] = [17] = \dots$

A **partition** of a set  $A$  is a collection of disjoint, nonempty subsets  $A_1, A_2, \dots, A_n$  whose union is all of  $A$ . For example, one possible partition of  $A = \{a, b, c, d, e\}$  is:

$$A_1 = \{a, c\} \qquad A_2 = \{b, e\} \qquad A_3 = \{d\}$$

These subsets are usually called the **blocks** of the partition.<sup>1</sup>

Here's the connection between all this stuff: there is an exact correspondence between *equivalence relations on  $A$*  and *partitions of  $A$* . We can state this as a theorem:

**Theorem 1.** *The equivalence classes of an equivalence relation on a set  $A$  form a partition of  $A$ .*

We won't prove this theorem (too dull even for us!), but let's look at an example. The congruent-mod-5 relation partitions the integers into five equivalence classes:

$$\begin{aligned} &\{\dots, -5, 0, 5, 10, 15, 20, \dots\} \\ &\{\dots, -4, 1, 6, 11, 16, 21, \dots\} \\ &\{\dots, -3, 2, 7, 12, 17, 22, \dots\} \\ &\{\dots, -2, 3, 8, 13, 18, 23, \dots\} \\ &\{\dots, -1, 4, 9, 14, 19, 24, \dots\} \end{aligned}$$

In these terms,  $x \equiv y \pmod{5}$  is equivalent to the assertion that  $x$  and  $y$  are both in the same block of this partition. For example,  $6 \equiv 16 \pmod{5}$ , because they're both in the second block, but  $2 \not\equiv 9 \pmod{5}$  because 2 is in the third block while 9 is in the last block.

In social terms, if "likes" were an equivalence relation, then everyone would be partitioned into cliques of friends who all like each other and no one else.

## 4 Partial Orders

A relation is a **partial order** if it is reflexive, antisymmetric, and transitive. In terms of properties, the only difference between an equivalence relation and a partial order is that the former is symmetric and the latter is antisymmetric. But this small change makes a big difference; equivalence relations and partial orders are very different creatures.

An example, the "divides" relation on the natural numbers is a partial order:

- It is reflexive because  $x \mid x$ .
- It is antisymmetric because  $x \mid y$  and  $y \mid x$  implies  $x = y$ .
- It is transitive because  $x \mid y$  and  $y \mid z$  implies  $x \mid z$ .

---

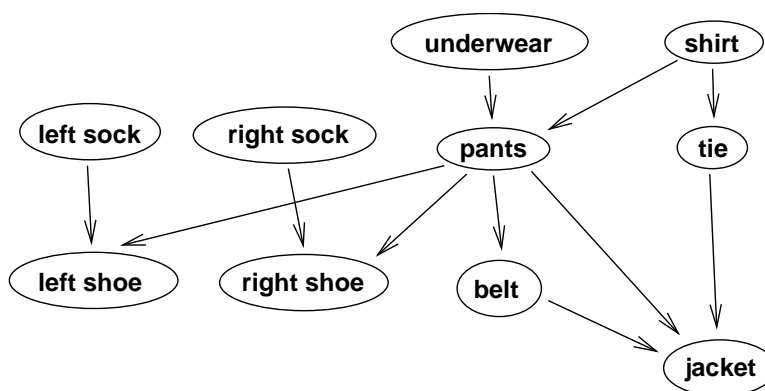
<sup>1</sup>I think they should be called the **parts** of the partition. Don't you think that makes a lot more sense?

The  $\leq$  relation on the natural numbers is also a partial order. However, the  $<$  relation is not a partial order, because it is not reflexive; no number is less than itself.<sup>2</sup>

Often a partial order relation is denoted with the symbol  $\preceq$  instead of a letter, like  $R$ . This makes sense from one perspective since the symbol calls to mind  $\leq$ , which is one of the most common partial orders. On the other hand, this means that  $\preceq$  actually denotes the *set* of all related pairs. And a set is usually named with a letter like  $R$  instead of a cryptic squiggly symbol. (So  $\preceq$  is kind of like Prince.)

Anyway, if  $\preceq$  is a partial order on the set  $A$ , then the pair  $(A, \preceq)$  is called a **partially-ordered set** or **poset**. Mathematically, a poset *is* a directed graph with vertex set  $A$  and edge set  $\preceq$ . So posets can be drawn just like directed graphs.

An example, here is a poset that describes how a guy might get dressed for a formal occasion:



In this poset, the *set* is all the garments and the *partial order* specifies which items must precede others when getting dressed. Not every edge appears in this diagram; for example, the shirt must be put on before the jacket, but there is no edge to indicate this. This edge would just clutter up the diagram without adding any new information; we already know that the shirt must precede the jacket, because the tie precedes the jacket and the shirt precedes the tie. We’ve also not bothered to draw all the self-loops, even though  $x \preceq x$  for all  $x$  by the definition of a partial order. Again, we know they’re there, so the self-loops would just add clutter.

In general, a **Hasse diagram** for a poset  $(A, \preceq)$  is a directed graph with vertex set  $A$  and edge set  $\preceq$  minus all self-loops and edges implied by transitivity. The diagram above is *almost* a Hasse diagram, except we’ve left in one extra edge. Can you find it?

Note that the directed graph of a poset that includes all of the self-loops and all the edges implied by transitivity is the **transitive closure** of the graph. The transitive closure of a relation  $R$  on  $A$  is a new relation  $R'$  on  $A$ , where  $R'$  contains all pairs  $(a_1, a_n)$ ,  $n \geq 1$  such that there is a “path” between  $a_1$  and  $a_n$  in  $R$ . By “path” we mean that exists some sequence

---

<sup>2</sup>Some sources omit the requirement that a partial order be reflexive and thus would say that  $<$  *is* a partial order. The convention in this course, however, is that a relation *must* be reflexive to be a partial order.

of elements  $a_1, a_2, \dots, a_n$  that are all related to one another by  $R$ :  $a_1 R a_2, \dots, a_{n-1} R a_n$ . For example, the transitive closure of the relation  $R = (1, 3), (1, 4), (2, 1), (3, 2)$  on the set  $A = 1, 2, 3, 4$  is  $R' = (1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4)$ .

## 4.1 Directed Acyclic Graphs

Notice that there are no **directed cycles** in the getting-dressed poset. In other words, there is no sequence of  $n \geq 2$  distinct elements  $a_1, a_2, \dots, a_n$  such that:

$$a_1 \preceq a_2 \preceq a_3 \preceq \dots \preceq a_{n-1} \preceq a_n \preceq a_1$$

This is a good thing; if there were such a cycle, you could never get dressed and would have to spend all day in bed reading books and eating fudgesicles. This lack of directed cycles is a property shared by all posets.

**Theorem 2.** *A poset has no directed cycles other than self-loops.*

*Proof.* We use proof by contradiction. Let  $(A, \preceq)$  be a poset. Suppose that there exist  $n \geq 2$  distinct elements  $a_1, a_2, \dots, a_n$  such that:

$$a_1 \preceq a_2 \preceq a_3 \preceq \dots \preceq a_{n-1} \preceq a_n \preceq a_1$$

Since  $a_1 \preceq a_2$  and  $a_2 \preceq a_3$ , transitivity implies  $a_1 \preceq a_3$ . Another application of transitivity shows that  $a_1 \preceq a_4$  and a routine induction argument establishes that  $a_1 \preceq a_n$ . Since we also know that  $a_n \preceq a_1$ , antisymmetry implies  $a_1 = a_n$  contradicting the supposition that  $a_1, \dots, a_n$  are distinct and  $n \geq 2$ . Thus, there is no such directed cycle.  $\square$

Thus, deleting the self-loops from a poset leaves a directed graph without cycles, which makes it a **directed acyclic graph** or **DAG**.

## 4.2 Partial Orders and Total Orders

A partially-ordered set is “partial” because there can be two elements with no relation between them. For example, in the getting-dressed poset, there is no relation between the left sock and the right sock; you could put them on in either order. In general, elements  $a$  and  $b$  of a poset are **incomparable** if neither  $a \preceq b$  nor  $b \preceq a$ . Otherwise, if  $a \preceq b$  or  $b \preceq a$ , then  $a$  and  $b$  are **comparable**.

A **total order** is a partial order in which every pair of elements is comparable. For example, the natural numbers are totally ordered by the relation  $\leq$ ; for every pair of natural numbers  $a$  and  $b$ , either  $a \leq b$  or  $b \leq a$ . On the other hand, the natural numbers are *not* totally ordered by the “divides” relation. For example, 3 and 5 are incomparable under this relation; 3 does not divide 5 and 5 does not divide 3. The Hasse diagram of a total order is distinctive:



A total order defines a complete ranking of elements, unlike other posets. Still, for every poset there exists some ranking of the elements that is consistent with the partial order, though that ranking might not be unique. For example, you can put your clothes on in several different orders that are consistent with the getting-dressed poset. Here are a couple:

underwear	left sock
pants	shirt
belt	tie
shirt	underwear
tie	right sock
jacket	pants
left sock	right shoe
right sock	belt
left shoe	jacket
right shoe	left shoe

A total order consistent with a partial order is called a “topological sort”. More precisely, a **topological sort** of a poset  $(A, \preceq)$  is a total order  $(A, \preceq_T)$  such that:

$$x \preceq y \quad \text{implies} \quad x \preceq_T y$$

So the two lists above are topological sorts of the getting-dressed poset. We’re going to prove that *every* finite poset has a topological sort. You can think of this as a mathematical proof that you *can* get dressed in the morning (and then show up for 6.042 lecture).

**Theorem 3.** *Every finite poset has a topological sort.*

We’ll prove the theorem constructively. The basic idea is to pull the “smallest” element  $a$  out of the poset, find a topological sort of the remainder recursively, and then add  $a$  back into the topological sort as an element smaller than all the others.

The first hurdle is that “smallest” is not such a simple concept in a set that is only partially ordered. In a poset  $(A, \preceq)$ , an element  $x \in A$  is **minimal** if there is no other element  $y \in A$  such that  $y \preceq x$ . For example, there are *four* minimal elements in the getting-dressed poset: left sock, right sock, underwear, and shirt. (It may seem odd that the minimal elements are at the top of the Hasse diagram rather than the bottom. Some people

adopt the opposite convention. If you're not sure whether minimal elements are on the top or bottom in a particular context, ask.) Similarly, an element  $x \in A$  is *maximal* if there is no other element  $y \in A$  such that  $x \preceq y$ .

Proving that every poset *has* a minimal element is extremely difficult, because this is actually false. For example the poset  $(\mathbb{Z}, \leq)$  has no minimal element. However, there is at least one minimal element in every *finite* poset.

**Lemma 4.** *Every finite poset has a minimal element.*

*Proof.* Let  $(A, \preceq)$  be an arbitrary poset. Let  $a_1, a_2, \dots, a_n$  be a maximum-length sequence of distinct elements in  $A$  such that:

$$a_1 \preceq a_2 \preceq \dots \preceq a_n$$

The existence of such a maximum-length sequence follows from the well-ordering principle and the fact that  $A$  is finite. Now  $a_0 \preceq a_1$  can not hold for any element  $a_0 \in A$  not in the chain, since the chain already has maximum length. And  $a_i \preceq a_1$  can not hold for any  $i \geq 2$ , since that would imply a cycle

$$a_i \preceq a_1 \preceq a_2 \preceq \dots \preceq a_i$$

and no cycles exist in a poset by Theorem 2. Therefore,  $a_1$  is a minimal element.  $\square$

Now we're ready to prove Theorem 3, which says that every finite poset has a topological sort. The proof is rather intricate; understanding the argument requires a clear grasp of all the mathematical machinery related to posets and relations!

*Proof.* (of Theorem 3) We use induction. Let  $P(n)$  be the proposition that every  $n$ -element poset has a topological sort.

*Base case.* Every 1-element poset is already a total order and thus is its own topological sort. So  $P(1)$  is true.

*Inductive step.* Now we assume  $P(n)$  in order to prove  $P(n+1)$  where  $n \geq 1$ . Let  $(A, \preceq)$  be an  $(n+1)$ -element poset. By Lemma 4, there exists a minimal element  $a \in A$ . Remove  $a$  and all pairs in  $\preceq$  involving  $a$  to obtain an  $n$ -element poset  $(A', \preceq')$ . This has a topological sort  $(A', \preceq'_T)$  by the assumption  $P(n)$ . Now we construct a total order  $(A, \preceq_T)$  by adding back  $a$  as an element smaller than all the others. Formally, let:

$$\preceq_T = \preceq'_T \cup \{(a, z) \mid z \in A\}$$

All that remains is the check that this total order is consistent with the original partial order  $(A, \preceq)$ ; that is, we must show that:

$$x \preceq y \quad \text{implies} \quad x \preceq_T y$$

We assume that the left side is true and show that the right side follows. There are two cases:

Case 1: If  $x = a$ , then  $a \preceq_T y$  holds, because  $a \preceq_T z$  for all  $z \in A$ .

Case 2: If  $x \neq a$ , then  $y$  can not equal  $a$  either, since  $a$  is a minimal element in the partial order  $\preceq$ . Thus, both  $x$  and  $y$  are in  $A'$  and so  $x \preceq' y$ . This means  $x \preceq'_T y$ , since  $\preceq'_T$  is a topological sort of the partial order  $\preceq'$ . And this implies  $x \preceq_T y$ , since  $\preceq_T$  contains  $\preceq'_T$ .

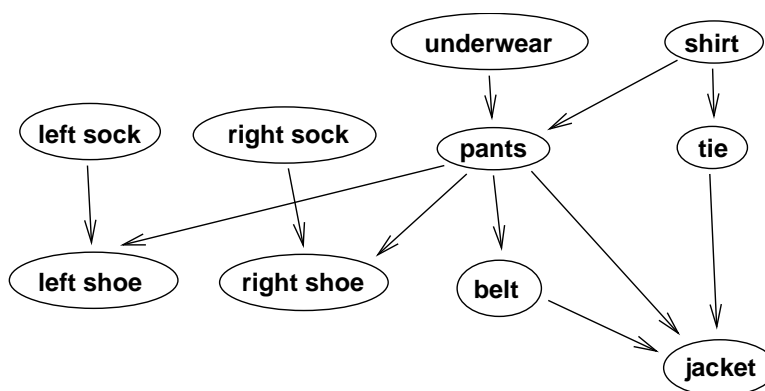
Thus,  $(A, \preceq_T)$  is a topological sort of  $(A, \preceq)$ . This shows that  $P(n)$  implies  $P(n+1)$  for all  $n \geq 1$ . Therefore,  $P(n)$  is true for all  $n \geq 1$  by the principle of induction, which prove the theorem.  $\square$

## 5 Parallel Task Scheduling

When items of a poset are tasks that need to be done and the partial order is a precedence constraint, topological sorting provides us with a legal way to execute the tasks sequentially - i.e., without violating 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, the partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one processor. How should we schedule the tasks? For simplicity, we will assume that all tasks take the same amount of time and that all processors are identical.

Assume all tasks take 1 unit of time and we have an unlimited number of identical processors. Given a poset of tasks, how long does it take to do all of them? For example, in the clothes example that we saw in lecture, how should we schedule the tasks?



In the first unit of time we should do all minimal items, so we would put on our left sock, our right sock, our underwear, and our shirt. In the second unit of time, we should put on our pants and our tie. Note that we cannot put on our left or right shoe yet, since we have not yet put on our pants. In the third unit of time we should put on our left shoe, our right shoe, and our belt. Finally, in the last unit of time we can put on our jacket.

The total time to do these tasks is 4 units. We cannot do any better than 4 units of time because there is a sequence of 4 tasks, each needing to be done before the next, of length

4. Here, for example, we must put on our shirt before our pants, our pants before our belt, and our belt before our jacket. Such a sequence of items is known as a *chain*.

**Definition 1.** A chain is a sequence  $a_1 \preceq a_2 \preceq a_3 \cdots \preceq a_t$ , where  $a_i \neq a_j$  for all  $i \neq j$ , such that each item is comparable to the next in the chain, and it is smaller with respect to  $\preceq$ .

Thus, the time it takes to schedule tasks, even with multiple processors, is at least the length of the longest chain. Indeed, if we used less time, then two items from the chain would have to be done at the same time, which contradicts the precedence constraints. For this reason, the longest chain is also known as a *critical path*.

Now in this example, we were in fact able to schedule all the tasks in  $t$  steps, where  $t$  is the length of the longest chain. The really nice thing about posets is that this is always possible! In other words, for any poset, there is a legal parallel schedule that runs in  $t$  steps, where  $t$  is the length of the longest chain.

There are a lot of ways to prove this fact. Our proof will also give us the corresponding schedule in  $t$  time steps, and allow us to obtain some nice corollaries.

**Theorem 5.** Given any finite poset  $(A, \preceq)$  for which the longest chain has length  $t$ , it is possible to partition  $A$  into  $t$  subsets  $A_1, A_2, \dots, A_t$  such that for all  $i \in \{1, 2, \dots, t\}$  and for all  $a \in A_i$ , we have that all  $b \preceq a$  appear in the set  $A_1 \cup \dots \cup A_{i-1}$ .

Before proving this theorem, first note that for each  $i$ , all items in  $A_i$  can be scheduled in time step  $i$ . This is because all preceding tasks are scheduled in preceding time steps, and thus are already completed. So the theorem implies that,

**Corollary 6.** The total amount of parallel time needed to complete the tasks is the same as the length of the longest chain.

*Proof.* (of Theorem 5) For all  $a \in A$ , put  $a$  in  $A_i$ , where  $i$  is the length of the longest chain ending at  $a$ . We show that for all  $i$ , for all  $a \in A_i$ , and for all  $b \preceq a$  with  $b \neq a$ , we have  $b \in A_1 \cup A_2 \cup \dots \cup A_{i-1}$ .

We prove this by contradiction. Assume there is some  $i$ ,  $a \in A_i$ , and  $b \preceq a$  with  $b \neq a$  and  $b \notin A_1 \cup A_2 \cup \dots \cup A_{i-1}$ . By the way we defined  $A_i$ , this implies there is a chain of length at least  $i$  ending at  $b$ . Since  $b \preceq a$  and  $b \neq a$ , we can extend this chain to a chain of length at least  $i + 1$ , ending at  $a$ . But then  $a$  could not be in  $A_i$ . This is a contradiction.  $\square$

If we have an unlimited number of processors, then the time to complete all tasks is equal to the length of the longest chain of dependent tasks. In recitation you'll see what we can do if there are only a limited number of processors.

It turns out that the theorem we just proved can be used to do a lot more than schedule tasks for parallel machines. First, we need a definition.

**Definition 2.** An antichain is a set of incomparable items, e.g., things that can be scheduled at the same time.

In our clothing example, underwear and shirt form an antichain. Another antichain is left sock, right sock, pants, and ties.

The above theorem can be recast in the language of antichains:

**Corollary 7.** *If  $t$  is the length of the longest chain in a poset  $(A, \preceq)$ , then  $A$  can be partitioned into  $t$  antichains.*

*Proof.* By the theorem, we can partition  $A$  into  $t$  sets, where each set consists of tasks that could be scheduled at the same time, i.e., incomparable items. Indeed, if some  $x \neq y$  were comparable, then either  $x \preceq y$  or  $y \preceq x$ , and  $x$  and  $y$  could not be scheduled at the same time.  $\square$