CHAPTER 1

Relations (section 4.1, 4.2 and 4.4)

1.1. Introduction

A mere set of words would not make a good dictionary.

It would be a pain to find a particular word!

A usable dictionary has some structure: the words are sorted.

In general, the more *structure* a set have, the more useful it is.

A way to bring structure into this set is to describe the *relations* between its elements, or between its elements and the elements of another set.

In this section we will see how we can formalize and study relations.

EXEMPLE. Imagine you want to build a house.

Figure 1.1.1 shows the tasks that need to be completed.

Let $S := \{F, W, E, I, O, R\}$ be the set of all tasks.

Problem: can we do the tasks in any order ?

For example, it would be better to build the walls AFTER the foundations!

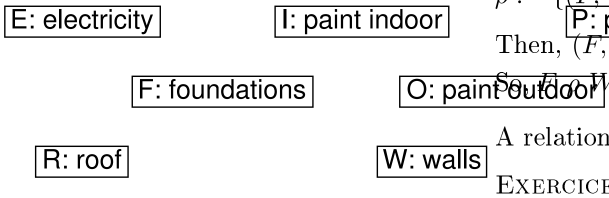


FIGURE 1.1.1. Tasks to build a house

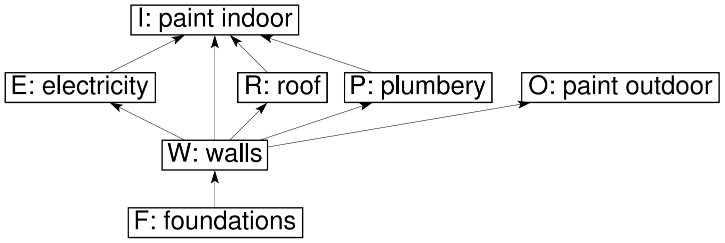


FIGURE 1.1.2. Constraints between the tasks to build a house

S in itself does not contain enough information to choose a correct order.

Set of constraints: $\rho := \{(F, W), (W, O), (W, E), (R, E), (R, I)\}$.

This set of constraints gives some structure to S, and makes it useful.

1.2. Relations

1.2.1. Definitions.

DÉFINITION. A *binary relation* on a set S is a subset ρ of $S \times S$.

Let x and y be two elements of S.

Then x is *in relation* with y (denoted $x\ \rho\ y$) iff $(x, y) \in \rho$.

EXEMPLE. Let ρ be the relation "is a prerequisite for":

$$\rho := \{(F, W), (W, O), (W, E), (R, E), (R, I)\}$$

Then, $(F, W) \in \rho$, but $(I, O) \notin \rho$.

So, $F\ \rho\ W$ is true, whereas $I\ \rho\ O$ is false.

A relation can be defined by a property.

EXERCICE 1. Let $S := \{1, 2, 3, 4, 5\}$. Draw the relations defined by:

(1) $x\ \rho_1\ y$ iff $x = y$;

(2) $x \ \rho_2 \ y$ iff $x \leq y$;
(3) $x \ \rho_3 \ y$ iff x divides y;
(4) $x \ \rho_4 \ y$ iff $x - y$ is even.

DÉFINITION. A *binary relation* between two sets S and T is a subset ρ of $S \times T$.

A *n-ary relation* between n sets $S_1, \dots, S_n$ is a subset ρ of $S_1 \times \cdots \times S_n$.

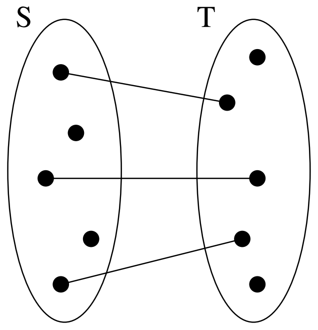
EXEMPLE. Let M be a set of male and F a set of female students.

We can define the relation "is married to".

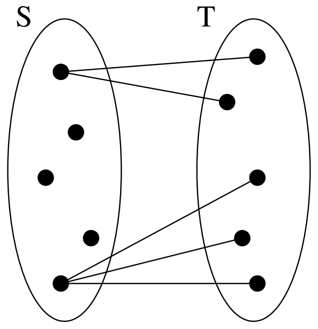
1.2.2. Basic properties of relations. Is there anything particular about the relation "is married to" ?

DÉFINITION. Let ρ be a relation between S and T.

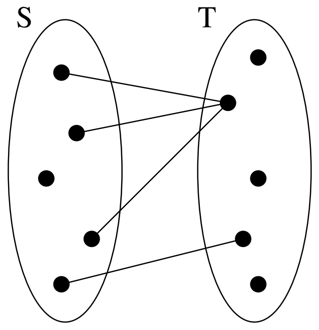
- S is *one-to-one* if any element of S and T appears at most once in ρ;



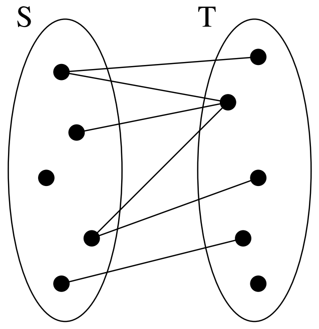
- S is *one-to-many* if any element of T appears at most once in ρ;



- S is *many-to-one* if any element of S appears at most once in ρ;



- S is *many-to-many* in all other cases.



EXEMPLE. Is x lower or equal to itself ?

DÉFINITION. A relation ρ on a set S is *reflexive* iff

$$(\forall x \in S) \ x \ \rho \ x.$$

EXEMPLE. Assume x is equal to y. Is y equal to x ?

DÉFINITION. A relation ρ on a set S is *symmetric* iff

$$(\forall x \in S)(\forall y \in S) \ (x \ \rho \ y) \leftrightarrow (y \ \rho \ x).$$

EXEMPLE. Assume $x \leq y$ and $y \leq x$. What can you say about x and y?

DÉFINITION. A relation ρ on a set S is *antisymmetric* iff

$$(\forall x \in S)(\forall y \in S)[(x \ \rho \ y) \text{ and } (y \ \rho \ x)] \rightarrow (x = y).$$

Exemple. Assume $x < y$ and $y < z$. What can you say about x and z?

Définition. A relation ρ on a set S is *transitive* iff

$(\forall x \in S)(\forall y \in S)(\forall z \in S)[(x\ \rho\ y) \text{ and } (y\ \rho\ z)] \rightarrow (x\ \rho\ z)$.

Exemple. What are the properties of the following relations

1. $<$, $\leq$, $=$;
2. "is married to";
3. "is a friend of";
4. "has the same color than".

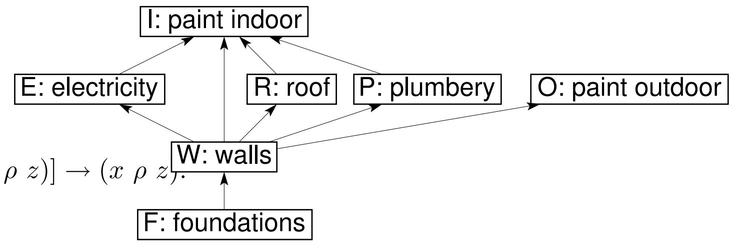


Figure 1.2.1. Constraints between the tasks to build a house

1.2.3. Operations on relations.

A relation is basically a set. So, we can use all usual set operations on relations.

Remarque. Let S and T be two sets.

The powerset $\wp(S \times T)$ is the set of all binary relations between S and T.

Définition. Let ρ and σ be two relations between S and T.

We can construct the following new relations:

- union: $\rho \cup \sigma$,
- intersection: $\rho \cap \sigma$,
- complement: ρ',
- ...

Exemple. Let $S := \mathbb{N}$.

1. What is the union of $<$and $=$?
2. What is the intersection of $<$ and $>$?
3. What is the union of $<$ and $>$?
4. Is $<$a sub relation of $\leq$?

1.2.4. Closure of a relation.

Exemple. Lets come back to the example of the house building.

Do you have to do the electricity after the foundations ?

Foundations is not a direct prerequisite for electricity.

However, it still needs to be done before electricity.

The relation "has to be done before" is transitive, and contains the relation "is a prerequisite for". It's the *transitive closure* of "is a prerequisite for".

Définition. Let ρ be a relation.

The *transitive closure* of ρ is the smallest transitive relation ρ^*containing ρ.

Exercice 2. Draw the transitive closure of "is a prerequisite for".

Problem 1.2.1. Is there an algorithm for computing the transivite relation of a relation?

Algorithme 1.2.2. *Construction of the transitive closure ρ^*.*

1. *$\rho^* := \rho$;*
2. *Find x, y, z such that $(x\ \rho^* y)$, $(y\ \rho^* z)$, and $(x\ \not\rho^* z)$;*
3. *If no such triple exists, $\rho^*$is transitive; Exit;*
4. *$\rho^* := \rho^* \cup \{(x, z)\}$;*
5. *Repeat at step 2.*

Remarque. The order in which you add the pairs to ρ is irrelevant.

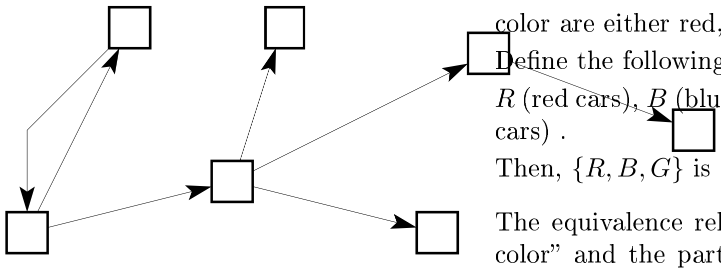
Exemple. Draw the relation "has to be done before" (Figure 1.4.1).

Définition. Let ρ be a relation.

The *reflexive closure* of ρ is the smallest reflexive relation containing ρ.

The *symmetric closure* of ρ is the smallest symmetric relation containing ρ.

Exercice 3. Consider the following relation:



(1) Draw its reflexive closure
(2) Draw its symmetric closure
(3) Draw its transitive closure
(4) Draw its antisymmetric closure

Remarque. The antisymmetric closure of a relation cannot be uniquely defined!

1.3. Equivalence relations

Exemple. Consider a set of cars, and the relation "has the same color as".

What are the properties of this relation?

Définition. An *equivalence relation* is a relation which is

(1) Reflexive
(2) Symmetric
(3) Transitive

Définition. Let ρ be a relation on a set S, and x be an element of S.

The *equivalence class* of x is the set $[x]$of all elements y such that $x\ \rho\ y$.

Exemple. The equivalence class of a car is the set of all cars having same color.

Remarque. If $x\ \rho\ y$, then $[x] = [y]$.

Exemple. Consider the relation $\equiv_3$ on $\mathbb{N}$ defined by $x \equiv_3 y$ iff 3 divides $x - y$.

(we also say that x and y are *congruent* modulo 3: $x \equiv y\ (mod\ 3)$)

What are the equivalence classes?

Définition. A partition of a set S is a collection of subsets $\{A_1, \ldots, A_k\}$ of S such that:

(1) $A_1 \cup \cdots \cup A_k = S$
(2) $A_i \cap A_j = \emptyset$ for any i, j.

Exemple. Consider a set S of car, whose color are either red, blue, or green.

Define the following subsets of S:

R (red cars), B (blue cars), and G (green cars) .

Then, $\{R, B, G\}$ is a partition of S.

The equivalence relation "has the same color" and the partition of the cars by color are closely related.

Théorème. *In general:*

(1) *An equivalence relation on S defines a unique partition of S.*
(2) *A partition of S defines a unique equivalence relation on S.*

Proof. Left as exercise:

(1) Construct the collection $\{A_1, \ldots, A_k\}$, and prove it is indeed a partition of S.
(2) Construct the relation ρ, and prove it's indeed an equivalence relation.

$\square$

Définition. Let S be a set, and ρ be a relation on S.

The set of all the equivalence classes is called the *quotient* of S by ρ.

Remarque. This method is used a lot in set theory:

Construction of $\mathbb{Z}$ from $\mathbb{N}$.

Construction of $\mathbb{Z}/n\mathbb{Z}$ (integers modulo n) from $\mathbb{Z}$.

Construction of $\mathbb{Q}$ from $\mathbb{Z}$.

1.4. Partially Ordered Sets

Exemple. What are the properties of the relation "has to be done before" on the tasks to build a house ?

Définition. A *partially ordered set* (or *poset*) is a set S with a relation ρ which is

(1) Reflexive
(2) Antisymmetric

(3) Transitive

Exemple. Which of the following are posets ?

(1) $(\{1,2,3,4\},\leq)$;
(2) $(\{1,2,3,4\},<)$;
(3) $(\{1,2,3,4\},=)$;
(4) $(\{1,2,3,4\},\rho)$, with $x\ \rho\ y$ iff x divides y.
(5) $(\wp(\{1,2,3,4\}),\subseteq)$.

1.4.1. Drawing posets; Hasse diagrams.

Définition. Some names:

- *Node* (or *vertex*)
- *Predecessor*
- *Immediate predecessor*
- *Maximal element*
- *Minimal element*
- *Least element*
- *Greatest element*

Définition. The *Hasse diagram* of a poset is a drawing of this poset such that:

- If $x\ \rho\ y$, then y is higher than x;
- The loops are not drawn;
- There is a segment linking x and y iff x is an immediate predecessor of y.

Exemple. Draw all posets on 1, 2, 3, 4 indistinct nodes.

How many such posets are there on 10 nodes ?

1.4.2. Scheduling. Scheduling a set of tasks consist in allocating tasks to resources, subject to certain constraints.

Exemple. Consider the house building problem, with the following assumptions:

- All tasks take the one day to complete;
- One worker can only work on one task every day;
- It does not save time to have two workers working on the same task.

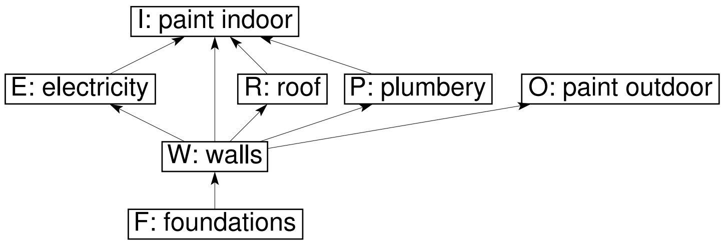


Figure 1.4.1. Constraints between the tasks to build a house

Schedule the tasks between 2 workers to optimize the completion time:

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Worker 1							
Worker 2							

What if there is one worker ?

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Worker 1							

What if there are three workers ?

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day7
Worker 1							
Worker 2							
Worker 3							

Proposition. *Consider the problem of scheduling n tasks on p processors, subject to precedence constraints, with the same assumptions and goal as above.*

- 1 or 2 processors: algorithms in $O(n^2)$;
- 3 processors: complexity unknown;
- p processors: NP-complete (basically the only known algorithm is exhaustive search);
- ∞ processors: $O(n)$.

This is typical from scheduling problems, where a very small change in the constraints can make huge differences in the complexity of the problems.

Also, usually the complexity is as follow:

- No constraints: low complexity;
- Some constraints: higher and higher complexity;
- More constraints: NP-complete. The only known algorithm is exhaustive search through all plausible cases (*branch-and-cut*);
- Even more constraints: still NP-complete, but easier. Indeed more and more cases can be thrown away very early.

Having very good scheduling algorithms is vital in industry, because a 1% difference in completion time (for example) can save thousands of dollars. Consequently, research in this topic is well financed!

1.5. Conclusion

- The more structure a set has, the more useful it is.
- Defining a relation is a way to add structure to a set.
- Relations, posets, equivalence relations, ... are just abstract models of natural notions commonly used in real life.