# CSC 202 Mathematics for Computer Science
# Lecture Notes

Marcus Schaefer
DePaul University[1]

# Chapter 2

# Databases and Set Theory

Recall the example asking for students in information systems and computer science. We saw that "and" here did not refer to a propositional and, but, instead meant we had to combine the results of two queries. We did this using a logical or in the condition. There is another way of thinking about it: we know how to get all computer science students:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE Program = 'COMP-SCI';
```

We also know how to get all information systems students, namely the same way:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE Program = 'INFO-SYS';
```

Could we just combine the results of the two queries instead of modifying the `WHERE` clause? The answer is yes, we can do so in SQL as follows:

```
(SELECT LastName, FirstName, SID
FROM Student
WHERE Program = 'COMP-SCI')
UNION
(SELECT LastName, FirstName, SID
FROM Student
WHERE Program = 'INFO-SYS');
```

We take the *union* of the two queries. Before we continue with this on the database side, let us first investigate the mathematical foundation of this notion: *set theory*.

## 2.1 Sets and Elements

Georg Cantor, the founder of set theory, defined what a set is as follows:

> A *set* is a collection of definite distinct objects of our intuition or of our thought into a whole. The objects are called the *elements* of the set.[1]

Slightly simplified: a set is a collection of elements. Let us have a look at some examples. Recall the Simpsons.[2] We could form a set containing the Simpsons as elements. We would write this set as follows:

$$S = \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\}.$$

So $S$ is a set containing five distinct elements. We use $\in$ for "is an element of". E.g.

$$\text{Homer} \in \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\},$$

or, to give another example,

$$7 \in \{2, 3, 5, 7, 11, \ldots\}.$$

On the other hand, we can express that something is *not* an element of a set using $\notin$, as in

$$\text{Selma} \notin \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\}.$$

We write $|X|$ for the *cardinality* of the set $X$, that is, the number of elements $X$ contains. So with $S$ as defined above, $|S| = 5$.

**Example 2.1.1.** Determining the cardinality of a set means counting the number of its elements. SQL allows you to do that too:

```
SELECT count(*)
FROM Student;
```

counts the number of students; count(*) means counting the number of records. Similarly,

```
SELECT count(*)
FROM Student, Enrolled, Course
WHERE SID = StudentID AND CourseID = CID AND
      Department = 'CSC' AND year = 2005;
```

[1] Or in the original: "Unter einer *Menge* verstehen wir jede Zusammenfassung $M$ von bestimmten wohlunterscheidbaren Objekten $M$ unserer Anschauung oder unseres Denkens (welche die *Elemente* von $M$ genannt werden) zu einem Ganzen."

[2] For those of you that do not watch TV, a commendable trait, "The Simpsons" is an American TV series centered around the Simpson family that, for our purpose, consists of Homer and Marge, the parents, and Lisa, Bart and Maggie, the children.

will give us the total enrollment in CSC courses in the year 2005. Note that this is not the same as the number of students enrolled in CSC courses in 2005, since a single student could have enrolled in multiple CSC courses and would therefore count multiple times. So, if we were asking how many students were enrolled in CSC courses in 2005, we would have to count SID, and, since SQL does not remove duplicates by default, we have to say we want to count *distinct* SID:

```
SELECT count(DISTINCT SID)
FROM Student, Enrolled, Course
WHERE SID = StudentID AND CourseID = CID AND
      Department = 'CSC' AND year = 2005;
```

Two sets are *equal* if they contain the same elements. In particular, the order of the elements is arbitrary (it just so happens that we have to write them down in a particular order when we do write them down). So, for example,

$$\{\text{Lisa}, \text{Bart}, \text{Maggie}\} = \{\text{Maggie}, \text{Lisa}, \text{Bart}\}.$$

By the same token, it does not matter whether we list an element several times, the set does not change:

$$\{\text{Lisa}, \text{Lisa}, \text{Lisa}\}$$

is the same set as $\{\text{Lisa}\}$, that is, it only contains *one* element, namely Lisa: $|\{\text{Lisa}, \text{Lisa}, \text{Lisa}\}| = 1$. If you are writing down a set (as an answer to a problem, say) you would never write something like $\{\text{Lisa}, \text{Maggie}, \text{Lisa}\}$ but write $\{\text{Maggie}, \text{Lisa}\}$ instead.

Elements relate to sets by the $\in$ or $\notin$ relation, two sets can be related by the *subset* relationship: $A$ is a subset of $B$ (written $A \subseteq B$) if every element of $A$ is also an element of $B$.[3]

So, for example,

$$\{\text{Maggie}, \text{Lisa}\} \subseteq \{\text{Maggie}, \text{Lisa}, \text{Bart}\},$$

and

$$\{\text{Lisa}, \text{Homer}\} \subseteq \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\},$$

while

$$\{\text{Maggie}, \text{Lisa}\} \not\subseteq \{\text{Maggie}, \text{Bart}\},$$

since Lisa is an element of $\{\text{Maggie}, \text{Lisa}\}$, but not of $\{\text{Maggie}, \text{Bart}\}$.

[3] You will sometimes read that $A$ is contained in $B$, but that is misleading, since this phrase is also used for elements, e.g. Homer is contained in $\{\text{Homer}, \text{Marge}\}$).

**Exercise 2.1.2.** What about the following containments: true or not?

$$\begin{aligned}
\{\text{Maggie}, \text{Lisa}\} &\stackrel{?}{\subseteq} \{\text{Lisa}, \text{Maggie}, \text{Bart}\} \\
\{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\} &\stackrel{?}{\subseteq} \{\text{Homer}, \text{Lisa}, \text{Bart}, \text{Maggie}\} \\
\{\text{Maggie}, \text{Lisa}\} &\stackrel{?}{\subseteq} \{\text{Lisa}, \text{Maggie}\} \\
\{\text{Maggie}, \text{Maggie}\} &\stackrel{?}{\subseteq} \{\text{Maggie}\} \\
\{\text{Maggie}, \text{Homer}, \text{Lisa}, \text{Marge}, \text{Bart}\} &\stackrel{?}{\subseteq} \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}\}
\end{aligned}$$

**Lemma 2.1.3.** *If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.*

Note: A *lemma* is the name mathematicians give to a small, useful result. Since it is a result, it requires a *proof*. When all the ingredients involved in the result are still very close to their definitions, one should go back to the definition to see why the result is true.[4]

*Proof.* We have to show that every element of $A$ is an element of $C$. However, if $x$ is an arbitrary element of $A$, then $x \in B$, because $A \subseteq B$. Then $x \in C$, since $B \subseteq C$. Which is what we had to show. ■

The result of the lemma (in a different guise, namely as a syllogism) goes back to Aristotelian logic, and, there, is known as *Modus Barbara*. You will often see it phrased as: if every $A$ is $B$ and every $B$ is $C$, then every $A$ is $C$. It can be nicely visualized (see Figure 2.1).

There is one special set that deserves mentioning, the *empty set*, the set not containing any elements whatsoever. It is typically written as $\{\}$ or, sometimes, $\emptyset$. The empty set is a subset of every set:

**Lemma 2.1.4.** *$\emptyset \subseteq B$ for every set $B$.*

*Proof.* By the definition of subset, $A \subseteq B$ is true if every element of $A$ is also contained in $B$. If $A$ is the empty set, i.e. if we are trying to verify that $\emptyset \subseteq B$, then we need to make sure that any element of $\emptyset$ is an element of $B$. Since $\emptyset$ does not have any elements, this is always true, whatever the set $B$ may be. ■

So, in particular $\emptyset \subseteq \{\text{Homer}, \text{Marge}\}$ and $\emptyset \subseteq \{1, 2, 3, \ldots\}$, and even $\emptyset \subseteq \emptyset$. You might well ask what good a set is that does not contain any elements. We will see several answers to this question later.

**Exercise 2.1.5.** Show that $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$. Does this remind you of anything?

[4] This is one possible reading of the first piece of advice given by Maass' *Grundriß der Logik*, 1836. See Appendix E for more information.

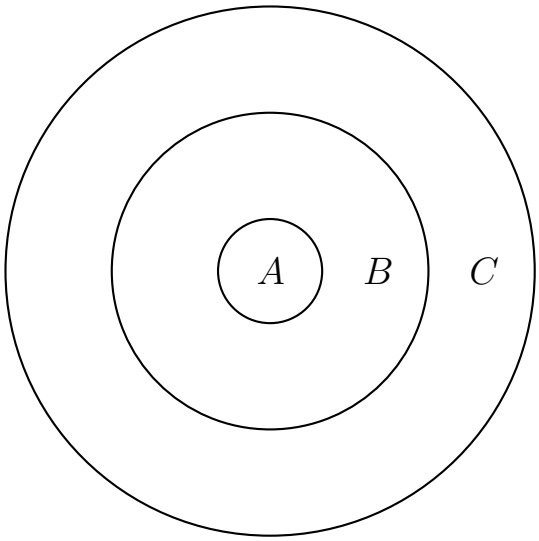


Figure 2.1: Modus Barbara

Before we continue to discuss operations on sets, we should make one general comment about where sets come from. The sample sets we have seen so far are given explicitly by listing all the elements in a set, e.g.

$$S = \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\},$$

or

$$A = \{1, 2, 3\}.$$

This stops being useful as the sets grow large or infinite. Imagine you want to talk about the set of all humans (which is tricky in itself, since that set keeps changing, and it can be hard to decide whether an element belongs to it or not, but let us ignore those difficulties for the moment). One way to capture the set is by listing all humans:

$$H = \{\text{Georg Cantor}, \ldots\}.$$

This is the *extensional* approach: defining a set by listing its members. This is the approach taken by databases (we will come back to this point later): we list all points of data we have explicitly. On the other hand of the spectrum, we could say that the set of humans is just that, the set of all humans:

$$H = \{x : x \text{ is human}\}.$$

This is the *intensional* way of defining a set: as a collection of all elements with a particular property ("being human" in this case). This, however, feels like cheating: we have only moved the emphasis from one grammatical category (the

noun "human") to another (the attribute "human"). However, the intensional approach is valid, for example, we could replace "being human" by definitions of being human. Here is one that is popular in the philosophical literature:

$$H = \{x : x \text{ is a featherless biped}\}.$$

Whether the extensional or the intensional point of view of a set is the right one depends very much on what you are using set theory for. For databases, the extensional point of view works best, for mathematics, the intensional point of view is often more useful. And for some areas, such as artificial intelligence, an intelligent combination of both is needed.

## 2.2 Nesting Queries using IN and EXISTS

Although we have not seen much set theory yet, we can already exploit some of it to write better SQL queries. Suppose we were interested in students in computer science, information systems, and computer gaming. We can easily extend one of the queries we saw earlier to get this result using a sequence of `OR`s. However, there is a more elegant way of doing it: SQL has an equivalent for the set theoretic $\in$ called `IN`, and we can write sets using parentheses instead of curly braces. So we can write

```
SELECT LastName, FirstName, SID
FROM Student
WHERE Program IN ('COMP-SCI', 'INFO-SYS', 'COMP-GAM');
```

which is much more readable than the equivalent

```
SELECT LastName, FirstName, SID
FROM Student
WHERE Program = 'COMP-SCI' OR Program = 'INFO-SYS' OR
      Program = 'COMP-GAM';
```

Suppose we are interested in all students that took "Theory of Computation" after 1995. Here is how we would currently write this query:

```
SELECT LastName, FirstName, Student.SID
FROM Student, Enrolled, Course
WHERE SID = StudentID AND CourseID = CID AND
      CourseName = 'Theory of Computation' AND year > 1995;
```

There is something slightly unsatisfactory about this query (from a database point of view) in that it mixes identifying the objects we are interested in with presenting the objects: for the former we only need the ID, for the later we need the names as well. If we were only interested in the IDs of students that took "Theory of Computation", we could have simplified the query to:

```
SELECT StudentID
FROM Enrolled, Course
WHERE CourseID = CID AND
      CourseName = 'Theory of Computation';
```

This is really the logical/computational heart of the query, the rest is presentation. Using `IN` we can nicely separate the presentation layer from the computational layer: we can write the full query as follows:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
   (SELECT StudentID
    FROM Enrolled, Course
    WHERE CourseID = CID AND
          CourseName = 'Theory of Computation');
```

Here is an added benefit of this new feature. Imagine you wanted to list all students that have not yet taken "Theory of Computation". We got stuck at that point earlier, since this cannot be solved by simply joining tables. Now it becomes as easy as moving from $\in$ to $\notin$, or, in SQL, from `IN` to `NOT IN`, namely

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID NOT IN
   (SELECT StudentID
    FROM Enrolled, Course
    WHERE CourseID = CID AND
          CourseName = 'Theory of Computation');
```

**Exercise 2.2.1.** Using `IN` and `NOT IN` (or both) solve the following problems.

1. List students that have taken courses in IT, GPH and DC.
2. (S) List presidents of student groups founded before 2000.
3. List students that have not taken any DC courses. (Verify your output. If it's different from what you are expecting, make sure you do the next exercise.)

**Exercise 2.2.2.** Compare the SQL query

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID NOT IN
   (SELECT StudentID
    FROM MemberOf, StudentGroup
    WHERE Name = GroupName AND
          Name IN ('HerCTI'));
```

to the SQL query

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
   (SELECT StudentID
    FROM MemberOf, StudentGroup
    WHERE Name = GroupName AND
          Name NOT IN ('HerCTI'));
```

They both seem to list students not in HerCTI. If you run them you will see that the outcomes are quite different. Carefully explain what each query actually *means* (in carefully phrased English). Then explain the difference.

Some queries can be more naturally phrased using SQL's `EXISTS` statement, which is equivalent to testing whether a set is not empty. For example, we could say

```
SELECT LastName, FirstName, SID
FROM Student
WHERE EXISTS (SELECT StudentID
              FROM Enrolled, Course
              WHERE CourseID = CID AND
                    StudentID = SID AND
                    CourseName = 'Theory of Computation');
```

That is, for each student, we check that there is a record of that student being enrolled in Theory of Computation. *That student* we express using `StudentID = SID`. We can use a fieldname from an outer query, `SID`, within an inner query. Such queries are often called *correlated* queries.

**Remark 2.2.3.** To better understand why the last query worked, we need to discuss how a database engine actually evaluates a query. It starts with the outer query (`FROM Student`) going through each record, one by one, verifying the `WHERE` condition. If there is a nested query, it then evaluates that nested query. It can be useful at this point, to use attributes of the record from the outer query, as we just saw.

**Exercise 2.2.4.**

1. (S) List all student groups that have a member (as per the information in the `MemberOf` table).
2. List graduate students enrolled in undergraduate courses and undergraduate students enrolled in graduate courses. (A course is an undergraduate course if its `CourseNr` is less than 420, otherwise it is a graduate course.)

SQL also offers `NOT EXISTS`, equivalent to testing whether a set is empty.

**Example 2.2.5.** Let us list all student groups that have no members. The members of a particular group, say HerCTI, we can list as

```
SELECT StudentID
FROM Memberof
WHERE Groupname = "HerCTI";
```

We now need to go through the `Studentgroup` table to check for which group this query comes back empty.

```
SELECT Name
FROM Studentgroup
WHERE NOT EXISTS (SELECT StudentID
                  FROM Memberof
                  WHERE Groupname = Name);
```

**Exercise 2.2.6.** 1. List all courses that nobody has ever enrolled in.

2. (S, Tricky) List student group presidents that are not members of the group they preside over.

3. List students that have never enrolled in a CSC course.

## 2.3 Basic Set Theory

The Simpson family consists of the parents $\{\text{Homer}, \text{Marge}\}$ and the children $\{\text{Lisa}, \text{Bart}, \text{Maggie}\}$. The Simpson family is the combination of those two sets, mathematically spoken, the union of those two sets, written as $A \cup B$:

$$\{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\} = \{\text{Homer}, \text{Marge}\} \cup \{\text{Lisa}, \text{Bart}, \text{Maggie}\}.$$

More formally, the *union* $A \cup B$ of two sets $A$ and $B$ is the set that contains all elements contained by either $A$ or $B$ (or, possibly, both).

**Exercise 2.3.1.** Write the Simpson set as the union of their male and their female elements.

Using what we know about logic, we can give a more formal definition of the union of two sets:

$$A \cup B = \{e : e \in A \ \vee e \in B\}.$$

If the two sets in $A \cup B$ do not have any elements in common, we say the sets are *disjoint*, we call $A \cup B$ the *disjoint union* of $A$ and $B$. Often a set gets split into two parts by whether a particular element fulfills a given requirement or not ("is male" versus "is female"; or "is parent" or "is child"). In those cases the union is always disjoint, which explains why we encounter this case quite often. For example, the natural numbers $\{1, 2, 3, \ldots\}$ are the disjoint union of the numbers that are even $\{2, 4, 6, 8, \ldots\}$ and those that are not $\{1, 3, 5, 7, \ldots\}$ (namely, the odd numbers).

$$\{1, 2, 3, \ldots\} = \{2, 4, 6, \ldots\} \cup \{1, 3, 5, \ldots\}.$$

Not all unions are disjoint, of course. For example,

$$\{\text{Lisa}, \text{Bart}\} \cup \{\text{Lisa}, \text{Maggie}\} = \{\text{Bart}, \text{Lisa}, \text{Maggie}\}.$$

If the union of two sets is not disjoint, those two sets must contain something in common. The *intersection*, $A \cap B$, of two sets is the set that contains all elements that belong to both $A$ and $B$; formally,

$$A \cap B = \{x : x \in A \wedge x \in B\}.$$

For example,

$$\{\text{Lisa}, \text{Bart}\} \cap \{\text{Marge}, \text{Lisa}\} = \{\text{Lisa}\},$$

and

$$\{\text{Marge}, \text{Homer}, \text{Maggie}\} \cap \{\text{Selma}, \text{Maggie}, \text{Marge}\} = \{\text{Marge}, \text{Maggie}\}.$$

**Exercise 2.3.2.**

1. (S) What is $\{\text{Marge}, \text{Lisa}\} \cap \{\text{Homer}, \text{Lisa}\}$?
2. What is $(\{\text{Marge}, \text{Homer}\} \cup \{\text{Lisa}, \text{Homer}\}) \cap \{\text{Maggie}, \text{Marge}, \text{Bart}\}$?
3. What is $(\{\text{Marge}, \text{Homer}\} \cap \{\text{Lisa}, \text{Homer}\}) \cup \{\text{Maggie}, \text{Marge}, \text{Bart}\}$

If two sets do not have any elements in common, their intersection $A \cap B$ is the empty set $\emptyset$. Here is one of the first important uses of the empty set: it is a way to state that two sets are disjoint: $A \cap B = \emptyset$.

Let us prove some simple facts about the notions we have seen so far:

**Lemma 2.3.3.** *The following statements are true for all sets $A$ and $B$:*

(*i*) $A \cap B = B \cap A$ *and* $A \cup B = B \cup A$,

(*ii*) $A \cap B \subseteq A$,

(*iii*) $A \subseteq A \cup B$,

(*iv*) *if* $A \subseteq B$, *then* $A \cap B = A$ *and* $A \cup B = B$

(*v*) *if* $A \cup B \subseteq A$, *then* $B \subseteq A$; *furthermore, if* $A \subseteq A \cap B$, *then* $A \subseteq B$.

Before we see the proof, consider the first statement: $A \cap B = B \cap A$. Take $A = \{\text{Marge}, \text{Homer}, \text{Lisa}\}$ and $B = \{\text{Lisa}, \text{Homer}, \text{Bart}\}$, then

$$A \cap B = \{\text{Lisa}, \text{Homer}\} = B \cap A,$$

so the statement in the lemma is true. Does that prove the statement of the lemma? No, it does not. It proves the statement is true for one particular choice of $A$ and $B$, but the lemma claims the statement is true for all choices of $A$ and $B$. How do we prove a general statement like this true? We cannot replace $A$ and $B$ with all possible sets $A$ and $B$. The answer is: we have to argue with what we know about sets, intersection, and equality of sets. That is, we take the intensional approach, based on an understanding of the concepts

involved, rather than the extensional approach. That should not distract from the importance of the extensional approach in math. Suppose somebody claimed

$$A \cup B = A \cap B \text{ for all sets } A \text{ and } B.$$

If you tried to prove this, you would quickly run into problems, indeed, you would probably find out that the statement is not true: take $A = \{\text{Lisa}, \text{Homer}\}$ and $B = \{\text{Lisa}, \text{Marge}\}$. Then

$$A \cup B = \{\text{Lisa}, \text{Homer}, \text{Marge}\} \neq \{\text{Lisa}\} = A \cap B.$$

The intended lesson is that before proving something in generality, first test it with some small examples, to make sure it really has a chance of being true.[5]

*Proof.* We will only prove two of the statements, the rest are left as exercises. First, let us show that the first part of $(i)$ is true, namely $A \cap B = B \cap A$ for all sets $A$ and $B$. If $x$ is an arbitrary element in $A \cap B$, it is an element of both $A$ and $B$. In other words, it is an element of both $B$ and $A$, and hence, an element of $B \cap A$. By the very same argument, an element of $B \cap A$ has to be an element of $A \cap B$, so $A \cap B$ and $B \cap A$ have the same element, and are, therefore, equal.

Second, let us prove the first part of $(v)$. We assume that $A \cup B \subseteq A$, so any element contained in either $A$ or $B$ has to belong to $A$. In particular, any element in $B$ has to belong to $A$, so $B \subseteq A$. ■

**Exercise 2.3.4.** Prove parts $(ii)$, $(iii)$, and $(iv)$ of Lemma 2.3.3.

**Remark 2.3.5.** We offer an alternative proof of the first part of $(i)$ in the lemma:

$$\begin{aligned} A \cap B &= \{x : x \in A \wedge x \in B\} \text{ by definition of } \cap \\ &= \{x : x \in B \wedge x \in A\} \text{ since } p \wedge q \text{ is equivalent to } q \wedge p \\ &= B \cap A \text{ by definition of } \cap \end{aligned}$$

The proof illustrates the close relationship between set theory and propositional logic: $\wedge$ corresponds to intersection, $\vee$ to union, and $\bar{\cdot}$ corresponds to complement (which we will see later). This allows you to translate any propositional equivalence into a set equality.

**Exercise 2.3.6.** What are the equivalents in set theory of the following propositional tautologies?

1. (S) $\varphi \wedge \varphi \leftrightarrow \varphi$,
2. $\varphi \vee \varphi \leftrightarrow \varphi$,
3. $\varphi \wedge (\psi \wedge \theta) \leftrightarrow (\varphi \wedge \psi) \wedge \theta$

[5] Recall Maass' 4th piece of advice to look at special cases.

4. $\varphi \wedge \psi \leftrightarrow \psi \wedge \varphi$
5. $\varphi \vee \psi \leftrightarrow \psi \vee \varphi$
6. (S) $\varphi \wedge (\psi \vee \theta) \leftrightarrow (\varphi \wedge \psi) \vee (\varphi \wedge \theta)$
7. $\varphi \vee (\psi \wedge \theta) \leftrightarrow (\varphi \vee \psi) \wedge (\varphi \vee \theta)$

When you see proofs that $A \cap B = B \cap A$ one reaction should be: but that is obvious. If it is not, then a good visualization of the sets and how they relate might help (you need to build a good mental model of the concepts you use; that will help you predict how they behave quickly rather than having to go through endless formalism; one could call this intuition). One such visualization is due to John Venn.[6] In Venn diagrams sets correspond to circles and what they enclose; for example, Figure 2.2 shows a Venn diagram for two sets. The diagram visually illustrates $A \cap B = B \cap A$.

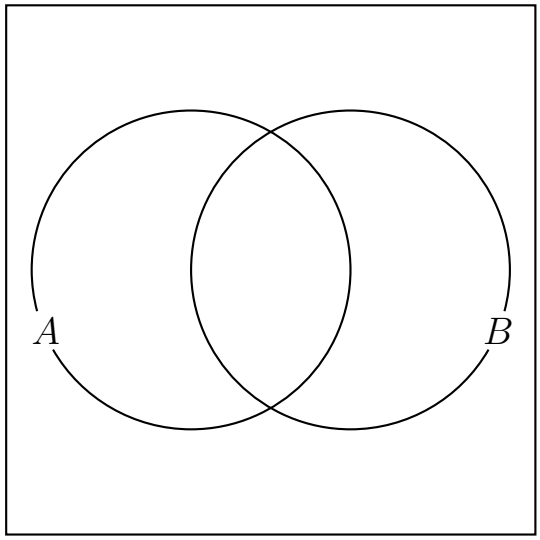


Figure 2.2: Venn diagram for $A$ and $B$.

Venn allowed different regions of the diagram to be shaded signifying that they were empty. For example, shading the part of $B$ outside of $A$ in Figure 2.2 would correspond to requiring that $B \subseteq A$.

Venn diagrams are a powerful way to visualize the relationship between different sets (or concepts or notions). You can build a whole logic on this (class logic), but we will not follow that trail here.

[6] English logician ... Much as we would like to discuss the history of logical diagrams, this is not the place. Let us just say that Venn diagrams are not as badly misnamed as some other notions in mathematics.

## 2.4 Basic Set Operations in Databases

Union is implemented by most database engines as `UNION`, as we saw in the opening example. To take the union of two `SELECT` statements they have to return the same type of table. For example, while we can write

```
(SELECT LastName, FirstName, SID
 FROM Student
 WHERE Program = 'COMP-SCI')
UNION
(SELECT LastName, FirstName, SID
 FROM Student
 WHERE Program = 'INFO-SYS');
```

it would *not* be legal to write

```
(SELECT LastName
 FROM Student
 WHERE Program = 'COMP-SCI')
UNION
(SELECT LastName, FirstName, SID
 FROM Student
 WHERE Program = 'INFO-SYS');
```

Intersection is implemented in a much smaller number of engines (Microsoft Access, for example, does not support it, H2, however, does). If it is supported it is typically called `INTERSECT`.

For example,

```
(SELECT LastName, FirstName, SID
 FROM Student
 WHERE Program = 'COMP-SCI')
INTERSECT
(SELECT LastName, FirstName, SID
 FROM Student
 WHERE Career = 'GRD');
```

lists all graduate students in computer science. The correct way of writing the query, however, would be a different one: you want to avoid doing operations on anything but primary keys, so really we should have written:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
      (SELECT SID
       FROM Student
       WHERE Program = 'COMP-SCI'
      INTERSECT
```

```
SELECT SID
FROM Student
WHERE Career = 'GRD');
```

This is quite certainly overkill in this particular example, but the point here is to illustrate the structure of the query.

In databases that do not support intersection, we can eliminate its use through propositional logic. A query equivalent to our last example would be:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
      (SELECT SID
       FROM Student
       WHERE Program = 'COMP-SCI') AND
       SID IN
      (SELECT SID
       FROM Student
       WHERE Career = 'GRD');
```

**Exercise 2.4.1.** 1. List students that are presidents of some student group and are enrolled in a CSC class.

2. List student groups which have both Chicago and non-Chicago students in them.

3. List courses that have been taken by both graduate students and undergraduate students (not necessarily in the same quarter).

## 2.5 Difference and Complement

We have seen unions and intersections of sets and how they relate to logical or and logical and. We have not found an equivalent for the logical not yet. The definition is straightforward. The *complement* of a set is the set of all elements *not* belonging to the set, i.e.

$$\overline{A} = \{x : x \notin A\}.$$

Or, is it? At a first glance this corresponds exactly to what we did when we transferred the notions of or and and to the realm of sets, but let us try to do an example: what is

$$\overline{\{\text{Maggie}\}} \ ?$$

By definition, $\overline{\{\text{Maggie}\}}$ contains all elements not in {Maggie}. And that does not just include Marge, Homer and the rest of the Simpsons, but all of the natural numbers and most of the universe. It seems that complementation is not a very useful notion. And, indeed, when mathematicians use the notion

of a complement of a set they always, implicitly, assume the complementation occurs within some limited world, often called the *universe of discourse*. For example, a number theorist interested in properties of the natural numbers, might reasonably write that the complement of the odd numbers are the even numbers:

$$\overline{\{1,3,5,7,\ldots\}} = \{2,4,6,8,\ldots\},$$

limiting the universe of discourse to the natural numbers, entirely excluding the Simpsons and other irrelevant stuff. Or we, in the Simpson family universe, might reasonably write

$$\overline{\{\text{Maggie}, \text{Bart}\}} = \{\text{Lisa}, \text{Homer}, \text{Marge}\},$$

excluding the natural numbers but also the rest of the Simpson world. What makes sense depends on your context and why you use the notion of complement.

**Lemma 2.5.1.** $A \cap \overline{A} = \emptyset$.

Using the Venn diagram for two sets $A$ and $B$ we can directly verify the truth of

$$\overline{A} \cap \overline{B} = \overline{A \cup B},$$

see Figure 2.3.

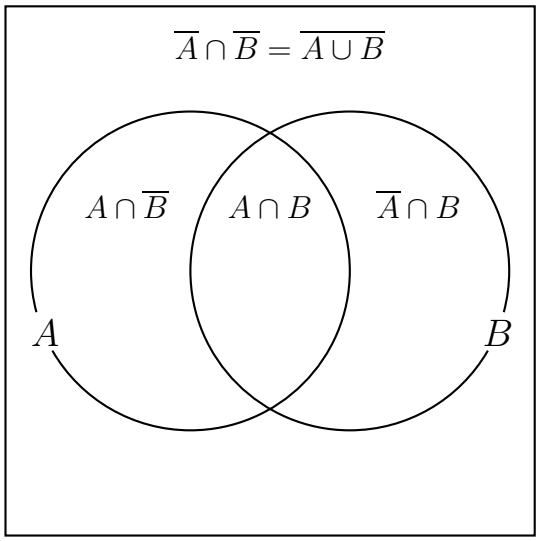


Figure 2.3: Venn diagram for $A$ and $B$ with complements.

The equality $\overline{A} \cap \overline{B} = \overline{A \cup B}$ should look slightly familiar: it closely resembles DeMorgan's law for propositions: $(\overline{p} \wedge \overline{q}) \leftrightarrow \overline{p \vee q}$. Indeed, we can use that law to prove the equality:

$$\begin{aligned}
\overline{A} \cap \overline{B} &= \{x : x \notin A \wedge x \notin B\} \\
&= \{x : \overline{(x \in A)} \wedge \overline{(x \in B)}\} \\
&= \{x : \overline{(x \in A \vee x \in B)}\} \text{ using DeMorgan's law} \\
&= \overline{\{x : x \in A \vee x \in B)\}} \\
&= \overline{A \cup B}
\end{aligned}$$

**Exercise 2.5.2.** Give a proof of the dual form, i.e.

$$\overline{A} \cup \overline{B} = \overline{A \cap B}.$$

As we observed earlier, every propositional equivalence corresponds to a set theoretic equality. Now that we have seen complementation, which corresponds to negation, we can translate some of the remaining equivalences we had.

**Exercise 2.5.3.** What are the equivalents in set theory of the following propositional tautologies?

1. $\varphi \leftrightarrow \overline{\overline{\varphi}}$,
2. (S) $\varphi \wedge \overline{\varphi} \leftrightarrow \bot$,
3. $\overline{\varphi} \wedge \overline{\psi} \leftrightarrow \overline{\varphi \vee \psi}$,
4. $\overline{\varphi} \vee \overline{\psi} \leftrightarrow \overline{\varphi \wedge \psi}$.

Databases do not implement the notion of complement, the potential universes of discourse being too large for this to make sense. Instead there is a limited form of complement resembling subtraction:

$$A - B := \{x : x \in A \wedge x \notin B\},$$

that is, we remove from $A$ all elements that are in $B$. Note that $A - B$ is the same as $A \cap \overline{B}$, collecting all elements in $A$ that are not in $B$.

We next prove a little lemma that will come in very handy later.

**Lemma 2.5.4.** *$A \subseteq B$ if and only if $A - B = \emptyset$.*

Before reading the more formal proof, look at the Venn diagram for $A$ and $B$ and convince yourself that the statement is true.

*Proof.* If $A \subseteq B$, then all elements of $A$ lie in $B$, so there are no elements of $A$ that are not in $B$, so the intersection of $A$ and $\overline{B}$, which is $A - B$, is empty.

On the other hand, if $A - B = \emptyset$, there are no elements belonging to both $A$ and $\overline{B}$. Hence, any element belonging to $A$ cannot belong to $\overline{B}$, and, therefore, must belong to $B$, which means that $A \subseteq B$. ■

## 2.6 Differences in SQL

A common type of query in a student database is prerequisite checking: when you enroll for a course, have you taken all the prerequisites required by that course. Let us consider one particular example: the Cryptography course requires that the student has previously taken the course "Java II". We want to list all students that are currently enrolled for the cryptography course without having taken Java II. Let us concentrate on getting the IDs of those students, we can then use the same set-up as earlier to get the full student information. So we want IDs of students enrolled in Cryptography, but not enrolled in Java II. Again, Maass is relevant. His third advice is that "often, the question can be split into several questions, which can [...] simplify finding the answer". The natural parts of our problem are: students currently enrolled in Cryptography, and students not enrolled in Java II. The first can be solved as follows (assuming that the current quarter is Fall 2007):

```
SELECT StudentID
FROM Enrolled, Course
WHERE CourseID = CID AND
      quarter = 'Fall' AND year = 2007 AND
      CourseName = 'Cryptography';
```

For the second query, we can again follow Maass, who in his sixth piece of advice suggests looking at the negation of a problem. So instead of looking for students *not* enrolled in Java II, let us look for students *not* not enrolled, that is, enrolled in Java II (recall that $\vdash p \leftrightarrow \overline{\overline{p}}$). However, students enrolled in Java II are easy to find:

```
SELECT StudentID
FROM Enrolled, Course
WHERE CourseID = CID AND
      CourseName = 'Java II';
```

What we need are the students from the first query without the students from the second query: if from all those students enrolled in Cryptography we remove those students that *have* taken Java II we are left with the students enrolled in Cryptography that have not taken Java II. This is exactly what set difference does for us. In SQL set difference is known as `EXCEPT` (though Oracle uses `MINUS`). So we could write

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
  (SELECT StudentID
   FROM Enrolled, Course
   WHERE CourseID = CID AND
         quarter = 'Fall' AND year = 2007 AND
```

```
        CourseName = 'Cryptography'
    EXCEPT
   SELECT StudentID
   FROM Enrolled, Course
   WHERE CourseID = CID AND
        CourseName = 'Java II');
```

Microsoft Access does not support `EXCEPT` (or `MINUS`), so we need to use a work-around if we actually want to run the query, by going back to propositional logic:

```
SELECT LastName, FirstName, SID
FROM Student
WHERE SID IN
   (SELECT Enrolled.SID
    FROM Enrolled, Course
    WHERE CourseID = CID AND AND
          quarter = 'Fall' AND year = 2007 AND
          CourseName = 'Cryptography')
    AND SID NOT IN
   (SELECT StudentID
    FROM Enrolled, Course
    WHERE CourseID = CID AND
          CourseName = 'Java II');
```

This will work in Access.

**Exercise 2.6.1.** 1. (S) List student groups that do not have any members.

2. List students that are not presidents of any student society.
3. List courses that nobody ever enrolled in.
4. List students which are enrolled only in CSC 489.
5. (Tricky) List students which took CSC 440, cryptography, without having taken "Java II" first. *Hint:* The difference to the example above is that this needs to work for any quarter in which CSC 440 is taught, not just the current one.

Let us tackle a slightly more difficult problem: List courses which only computer science students ever enrolled in.

Maass' 4th piece of advice comes in handy here: "it is often helpful to consider a special, more particular, version of the question". So, let us look at a particular course, let us say CSC 440 Cryptography. By hand, we can check that there was a non-computer science student in the course, so the course should not be listed.

How did we make this decision? What did we compare? (Think about this, maybe redoing the example, before reading on.) Let $A$ be the set of students

enrolled in CSC 440 and let $B$ be the set of all computer science students. We need to verify that $A \subseteq B$. As we saw earlier, this condition is equivalent to $A - B = \emptyset$ (Lemma 2.5.4). And we know how to test for the empty set in SQL by using `NOT EXISTS`. So our strategy is clear now: we need to build queries for $A$ and $B$, and use `NOT EXISTS` to test whether $A - B$ is empty. Here is the query for $A$:

```
SELECT StudentID
FROM Enrolled
WHERE CourseID = '1092';
```

$B$ is also easy:

```
SELECT SID
FROM Student
WHERE Program = 'COMP-SCI';
```

So we can test whether there were any computer science students in CSC 440 in fall of 2005 as follows:

```
NOT EXISTS (
    SELECT StudentID
    FROM Enrolled
    WHERE CourseID = '1092'
   EXCEPT
    SELECT SID
    FROM Student
    WHERE Program = 'COMP-SCI');
```

If this condition is true, then all students in CSC 440 in fall of 2005 are computer science students. Now, instead of this particular course, we want to check all courses. So we need an outer query running through all courses.

```
SELECT Department, CourseNr, CourseName
FROM Course
WHERE NOT EXISTS (
    SELECT StudentID
    FROM Enrolled
    WHERE CourseID = CID
   EXCEPT
    SELECT SID
    FROM Student
    WHERE Program = 'COMP-SCI');
```

**Remark 2.6.2.** Here is an alternative way of solving the problem. Why does it lead to the same result?

```
SELECT Department, CourseNr, CourseName
FROM Course
WHERE  0 = (SELECT count(*)
            FROM Student
            WHERE SID IN (
               SELECT StudentID
               FROM Enrolled
               WHERE CourseID = CID
              EXCEPT
               SELECT SID
               FROM Student
               WHERE Program = 'COMP-SCI'));
```

**Exercise 2.6.3.** 1. (S) List students which have enrolled in CSC courses only (you can include students that have not enrolled in any courses).

2. List courses that were not taught before 2006.
3. List courses that only graduate students have enrolled in.
4. List student groups all of whose members are graduate students.
5. List all courses that have been taught every year (that courses have been taught).
6. List all courses that have been taught every quarter (that courses have been taught).

**Remark 2.6.4.** The request "List courses which only computer science students ever enrolled in" was not very realistic; we would have preferred to ask for courses that only computer science students enrolled in, where we distinguish an "abstract" course, like CSC 440 from an actual offering of the course, like the one in fall of 2005. Our database is not very well set up for this (typically you have different tables for courses and sections of courses, which we do not), but it can be done; at the expense of introducing a new feature: naming. Occasionally, you have to use the same table twice, which leads to confusion when you refer to a field by name, since it is not clear which of the two tables you want the field to come from. Hence, you give the tables names as you introduce them. For example, suppose we were looking for different students with the same name, we could write

```
SELECT A.LastName, A.SID, A.FirstName, B.SID, B.FirstName
FROM Student AS A, Student AS B
WHERE A.LastName = B.Lastname AND
      A.FirstName = B.FirstName AND
      NOT (A.SID = B.SID);
```

using `AS` to obtain two named versions of the student table. We can use the same tool to distinguish two different occurrences of the `Enrolled` table, which we need to distinguish different offerings of the same class.

```
SELECT DISTINCT Department, CourseNr, CourseName, E1.Quarter, E1.Year
FROM Enrolled AS E1, Course
WHERE E1.CourseID = CID AND
   NOT EXISTS (
    (SELECT E2.StudentID
    FROM Enrolled AS E2
    WHERE E2.CourseID = CID AND
          E2.Quarter = E1.Quarter AND
          E2.Year = E1.Year)
   EXCEPT
    (SELECT SID
    FROM Student
    WHERE Program = 'COMP-SCI'));
```

## 2.7 Pairs and Tuples and Relations

A pair of two objects $x$ and $y$ is written as $(x, y)$. Its main characteristic is that it is ordered: a pair has a first component and a second component. So for two pairs $(x, y)$ and $(u, v)$ to be equal we need to have $x = u$ and $y = v$, that is, the pairs are equal component by component.

**Exercise 2.7.1.** Explain why $(x, y)$ is not the same as $\{x, y\}$.

Given two sets $X$ and $Y$ we can form the *Cartesian product* of the two sets by collecting all pairs:

$$X \times Y := \{(x, y) : x \in X \wedge y \in Y\}.$$

**Example 2.7.2.** Let $X = \{\text{Homer}, \text{Marge}, \text{Lisa}, \text{Bart}, \text{Maggie}\}$. Then $X \times X$ consists of $|X \times X| = 25$ elements, including $(\text{Homer}, \text{Homer})$, $(\text{Homer}, \text{Marge})$, ..., $(\text{Maggie}, \text{Maggie})$.

We can take the product of more than two sets by building *tuples*, instead of pairs. For example,

$$X \times Y \times Z = \{(x, y, z) : x \in X, y \in Y, z \in Z,$$

is the set of all *triples* with first component in $X$, second component in $Y$, and third component in $Z$. If we have $n$ sets $X_1, \ldots, X_n$, their Cartesian product is

$$X_1 \times X_2 \times \ldots \times X_n = \{(x_1, x_2, \ldots, x_n) : x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n\}.$$

Tuples in general have the same defining characteristic as pairs:

$$(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$$

if and only if $x_i = y_i$ for all $1 \leq i \leq n$.

The Cartesian product is named after René Descartes, whose name should be familiar from *Cartesian coordinates*: every point in the Euclidean plane can

we written as $(x, y)$, where $x$ is the $x$-coordinate of the point and $y$ is the $y$-coordinate. In other words, if $\mathbb{R}$ is the real line, then Descartes claimed that the Euclidean plane is simply $\mathbb{R} \times \mathbb{R}$. Euclidean space is $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$, i.e. a triple consisting of the $x$, $y$, and $z$-coordinate of each point.

You should already have gotten an inkling why cartesian products are relevant to databases: a record in a database corresponds to a tuple, with its coordinates belonging to the domain of its field (an integer, a string, etc.). For example, the table `StudentGroup` is created as:

```
create table studentgroup (
  Name  varchar(40),
  PresidentID   int(5),
  Founded       year,

  primary key (Name)
);
```

That is, `Name` is a string of length at most 40, `PresidentID` is an integer with at most 5 digits, and `Founded` is a year (which is a special type). So if we let $A$ be the set of all strings of length at most 40, $B$ be the set of integers with at most 5 digits, and $C$ be the set of all years, then a record in this table is an element of $A \times B \times C$.

Indeed, taking this one step further, any particular state of the `Studentgroup` table is simply a subset of $A \times B \times C$. Of course, the same is true for all tables and relations in the relational database world: they are subsets of the Cartesian products of the domains of their fields.

---

## Excursion: Databases and Tuples

As we just saw, a `SELECT` query returns a list of tuples. We can use this in conjunction with `IN` to simplify some queries. For example, suppose we want to list the information about CSC 440, IT 240 and IT 223 (you will often encounter strange lists like that). We already know how to do this, but here is a more elegant way.

```
SELECT CID, Department, CourseNr
FROM Course
WHERE (Department, CourseNr) IN (('CSC', '440'), ('IT', '240'), ('IT', '223'));
```

Here is another example: suppose we want to list all students that joined 'DeFrag' in 2004 or 'HerCTI' in 2005. We can now write this as:

```
SELECT *
FROM Student, Memberof
WHERE StudentID = SID AND
    (GroupName, Joined) IN (('DeFrag', 2004), ('HerCTI', 2005));
```

(This query does not work in H2, there seems to be a type problem.)

The list of pairs (or other tuples) can also be the result of a `SELECT` query. For example, let us list students that took a course during the year that they started:

```
SELECT *
FROM Student
WHERE (SID, Started) IN
      (SELECT StudentId, year
       FROM Enrolled);
```

(Again, this query doesn't work in H2.)

---

## 2.8 More Sets

If a point in the plane is an element of $\mathbb{R} \times \mathbb{R}$, then a geometric figure, a set of such points, is a subset of $\mathbb{R} \times \mathbb{R}$, and the set of all geometric figures is the set of all subsets of $\mathbb{R} \times \mathbb{R}$.

If a table in a database is a subset of the Cartesian product of its domains, then the set of all possible tables is the set of all subsets of that Cartesian products. You can keep multiplying examples all making the point that it is useful to be able to talk about all subsets of a set. This is known as the *powerset*:

$$\mathbf{P}(A) = \{X : X \subseteq A\}.$$

**Example 2.8.1.** With $S = \{\text{Marge}, \text{Homer}\}$, we have

$$\mathbf{P}(S) = \{\emptyset, \{\text{Marge}\}, \{\text{Homer}\}, \{\text{Marge}, \text{Homer}\}\}.$$

Note that the empty set and the whole set itself always occur in the powerset, since they are the trivial subsets of the set.

**Exercise 2.8.2.** Compute $\mathbf{P}(\{\text{Marge}, \text{Homer}, \text{Bart}\})$. Then compute $\mathbf{P}(\emptyset)$. At that point try to make a guess at $|\mathbf{P}(S)|$ for finite sets $S$.

Doing some examples should have convinced you that

$$|\mathbf{P}(S)| = 2^{|S|}$$

for finite sets: subsets differ by which elements they contain and each element can either be contained or not, giving us two choices per element, yielding

$$\overbrace{2 * 2 * \cdots * 2}^{|S|} = 2^{|S|}$$

different subsets.

Since $2^n > n$ for all $n$, the powerset of a set is always larger than the set itself; amazingly, this is still true for infinite sets as we will see later.

**Exercise 2.8.3.**

1. How many elements does $\mathbf{P}(\{\text{Marge, Homer, Lisa, Bart, Maggie}\})$ have?
2. Compute $\mathbf{P}(\{1,2,3\})$ and draw the result as a diagram which naturally shows how the subsets are included in each other; start with $\emptyset$ at the bottom and $\{1,2,3\}$ at the top, and include the remaining subsets of $\{1,2,3\}$ between them; draw an arrow from one subset $A$ to another subset $B$ if $A \subseteq B$. Try to arrange the sets so that the subset relationship is reflected visually.
3. Compute $\mathbf{P}(\{1,2,3,4\})$ and draw the result as a diagram which naturally shows how the subsets are included in each other; start with $\emptyset$ at the bottom and $\{1,2,3,4\}$ at the top, and include the remaining subsets of $\{1,2,3\}$ between them; draw an arrow from one subset $A$ to another subset $B$ if $A \subseteq B$. Try to arrange the sets so that the subset relationship is reflected visually.
4. In the diagrams you did for the previous two exercises what does taking the union or intersection of two sets correspond to?

The definition of powerset also works for infinite sets, for example, $\mathbb{N} \in \mathbf{P}(\mathbb{R})$, or $\{2,3,5,7,\ldots\} \in \mathbf{P}(\{1,2,3,\ldots\})$.

Every $A \in \mathbf{P}(\{1,2,3,\ldots\})$ contains a smallest element.

Looks like a very simple statement, but, in truth, is the most powerful tool we have for dealing with natural numbers. It is known as the principle of mathematical induction. You can tell that it is special, since it already fails to be true for $\mathbb{R}$: not every element in $\mathbf{P}(\mathbb{R})$ has a smallest element (example?).

## 2.9 *Paradoxes of Set Theory

What we have seen of set theory so far all belongs to a theory more accurately called *naïve set theory*. It is a very useful tool (as we have seen, and as we will see), but from a strictly mathematical point of view it has a major problem: it contains a self-contradiction, and the one thing mathematics should not be is contradictory.

The contradiction takes the form of a paradox: in naïve set theory there is a statement whose truth implies its falsehood and whose falsehood implies its truth, so apparently it cannot be either true or false.

Maybe the most elegant form of the paradox, and also the first published one, is due to Bertrand Russell[7] who considered the set of all sets that do not contain themselves. More formally (and paradoxes are always a good indication that one should get a bit more formal), let

$$R = \{x : x \notin x\}.$$

[7] Of course, there are claims to priority by other logicians.

This set looks a bit strange: it consists of other sets, namely of those that do not contain themselves. Are there sets that contain themselves? Well, yes; if you admit, for example, the set of *all* sets, called $V$, then $V \in V$, since, as a set, it must contain itself. Most sets, however, do not contain themselves. What about $R$ though? If $R \in R$, then $R \notin R$ is false, so, by the way $R$ was defined, $R \notin R$. So assuming that $R \in R$ is true, leads to the conclusion that it is false. Similarly, if we assume that $R \notin R$, then again just following the way $R$ was defined, $R$ should contain itself: $R \in R$, again in contraction to the assumption we started with.

So assuming either that $R$ belongs to $R$ or the opposite leads to a contradiction. So, which one is it?[8]

## 2.10 Exercises

1. Show that $|A \cup B| = |A| + |B| - |A \cap B|$.
2. Show that $|A \triangle B| = |A| + |B| - 2|A \cap B|$, where $A \triangle B = (A - B) \cup (B - A)$ is the *symmetric difference* of $A$ and $B$.
3. What does it mean if the symmetric difference of two sets is empty? (See previous exercise for the definition of symmetric difference.)
4. Is it true that $(A - (B - C)) = ((A - B) - C)$? That is, is set difference associative? If it is, prove it, if not, find a counterexample.
5. Use shading in a Venn diagram for two sets to express that $A \cap B = \emptyset$.
6. Use shading in a Venn diagram for two sets to express that $A \cap \overline{B} = \emptyset$. What does that mean?
7. Use shading in a Venn diagram for two sets to express that $\overline{A} \cap \overline{B} = \emptyset$. What does that mean?
8. Draw a Venn diagram for three sets $A$, $B$ and $C$ and label all the (eight) regions with the sets they correspond to (e.g. $A \cap B \cap C$ or $\overline{A} \cap B \cap \overline{C}$).
9. Using shading, draw Modus Barbara (Figure 2.1) as a special case of the Venn diagram for three regions.
10. Use shading in a Venn diagram for three sets to express that $A \cap \overline{C} = \emptyset$, $B \cap \overline{C} = \emptyset$, $\overline{A} \cap \overline{B} \cap C = \emptyset$. What does that mean?
11. We introduced the pair $(a, b)$ of two elements $a$ and $b$ as a primitive notion. Show that it can be defined using sets. *Hint:* The first attempt would be to define $(a, b)$ as $\{a, b\}$. That, however, does not work, since $\{a, b\} = \{b, a\}$ but we not want $(a, b) = (b, a)$ in general.

[8] Philosophers have taken all possible positions on this: the statement is true, the statement is false, its both, or its neither. You have your pick.