

Course Outline

- Logic
- Proof techniques
- Complexity
- Floating-point number systems

Motivation

This course is designed to help you think like a computer scientist.

The main focus is **communicating precisely**:

- knowing and saying exactly what we mean, and
- understanding exactly what others say and mean.

In particular, understanding what it means to understand.

This course will help you read technical material:

- during your university career (course textbooks), and
- during your professional career (journals and books containing cutting edge research in AI, cryptography, databases, graphics, networks, etc.

Mathematics vs. Computer Science

Computer science is a mathematical science.

It uses techniques of mathematics, and material from branches of mathematics:

- Computer graphics: multivariable calculus, physics-based modeling
- Digital signal processing: multivariable calculus (e.g., speech understanding and synthesis)
- Numerical analysis: multivariable calculus, linear algebra
- Cryptography: number theory
- Networking: graph theory, statistics
- Algorithms: combinatorics, probability, set theory
- Databases: set theory, logic
- AI: set theory, logic
- Programming languages: set theory logic
- Formal methods: set theory and logic for the specification and verification of hardware and software; nuclear, aviation (NASA), hardware design (INTEL), and other industries use it.

We will use both computing-related and mathematical-based examples.

Who should take this course?

Do you memorize math? **YES**

Do you have trouble explaining what you are doing in a mathematical question? **YES**

Do you have trouble understanding what word problems are asking? **YES**

Do you like reading math textbooks to learn new math? **NO**

Are you just as happy talking about abstract things x and y , as about specific concrete examples? And vice versa? **NO**

Have you passed CSC238? **NO**

How to succeed in this course

- Practice - you are learning a new language and it you you can't just cram before an exam.
- Ask questions
- Read the textbooks
- Apply what you are learning to your math course(s)

Precision

Programming: expressing yourself in a very constrained language

Human communication: lots of context, intuition to remove ambiguities

Technical communication: need additional precision

Being precise allows us to make strong claims, e.g.,

(1) Differential functions are continuous.

Compare that with:

(2) Dogs are furry.

What is the difference?

(2) needs more precision, perhaps “Dogs are usually furry”. Or maybe breeds like Mexican hairless are not considered “real” dogs, so the concept of “dog” needs to be defined more precisely.

Once the meaning is clear, we can discuss truth.

Does every person agree with (2)?

Does every mathematician agree with (1)?

Mathematicians and computer scientists talk precisely about precise concepts in their work. But precision also depends on the audience. For example:

(3) `/** Sorts a in ascending order */`
`public void sort(int[] a) ...`

(4) `// sets a to 1`
`a = 1;`

In (3), “a” means “the object referred to by the value in a”, whereas in (4), “a” means “the variable a”.

Lack of precision can be a problem for people who are unfamiliar with Java. But Java programmers need to rely on the audience familiar with programming to avoid typing and read long comments.

Warning! In this course, we can't assume you know what you're saying, so you have to demonstrate it explicitly. At the same time, you have to learn to understand statements that are seemingly imprecise, but that can be made precise from the context.

Universal Quantification

Basic type of claim:

certain things have certain properties

For example, consider a database of employees and their salaries:

| Employee | Salary |
|----------|--------|
| Anya | 57000 |
| Dawn | 0 |
| Giles | 40000 |
| Willow | 30000 |
| Xander | 50000 |
| Buffy | 18000 |

Now consider the claim:

Every employee makes less than 60000.

Is it true? [Yes, check each employee.]

This is an example of “[universal quantification](#)”, which is indicated by “every”, “each” or “all”.

Here are some more examples of “universal quantification”:

- Each employee makes less than 60000.
- All employees make less than 60000.

All of these are considered to be the exact same claim. Sometimes only plural is used:

Employees make less than 60000.

Recall the “dogs are furry” example. Let’s be precise: universal quantification really means [every](#), with no exceptions.

Next we’ll consider the claim:

Each employee makes at least 10000.

Is it true? [No: Dawn doesn’t make at least 10000.]

Dawn is called a “counterexample” to/for the claim.

Let’s add a column to our database:

| Employee | Gender | Salary |
|----------|--------|--------|
| Anya | F | 57000 |
| Dawn | F | 0 |
| Giles | M | 40000 |
| Willow | F | 30000 |
| Xander | M | 50000 |
| Buffy | F | 18000 |

Here is another claim:

(5) All male employees make less than 55000.

Is (5) true? [Yes: restrict to male employees, check each one. Careful: do not restrict to employees making less than 55000.]

Compare that with the claim:

(6) Every female employee makes less than 55000.

Is (6) true? [No: Anya is a counterexample.]

Note: to disprove claims like (5) and (6), a single counterexample is enough.

In general, to show that a universally quantified statement, such as
all Ps are Qs

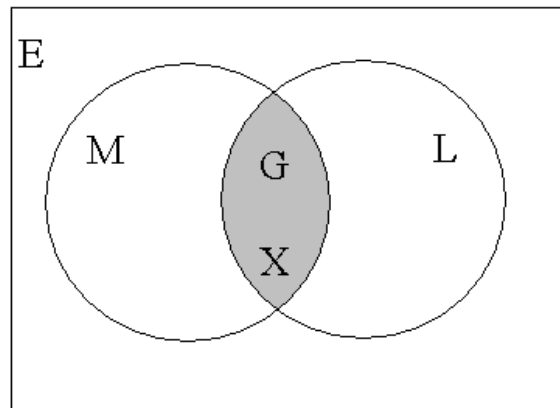
is false, it is enough to give one counterexample (a P that is not a Q).

Properties as sets

It is useful to view properties as sets and vice versa.

Consider (5) again in terms of sets:

- E: Employees
- M: Male employees
- L: Employees making less than 55000



Now, (5) can be rephrased:

Everything in M is (also) in L.

In other words, “M is a subset of L”.

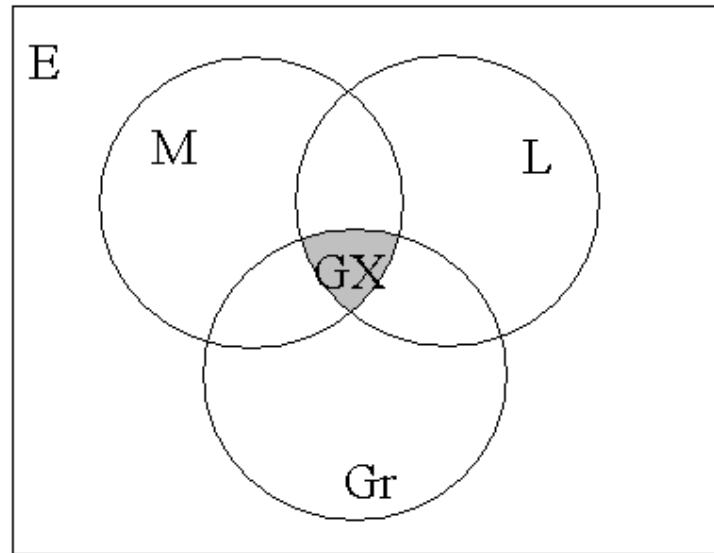
This can be represented symbolically as: $M \subseteq L$

We need to be precise about the meaning of “subset”. Subset may also mean that “M is equal to L”, so subset does not mean “proper subset”.

For example:

All men make between 35000 and 55000.

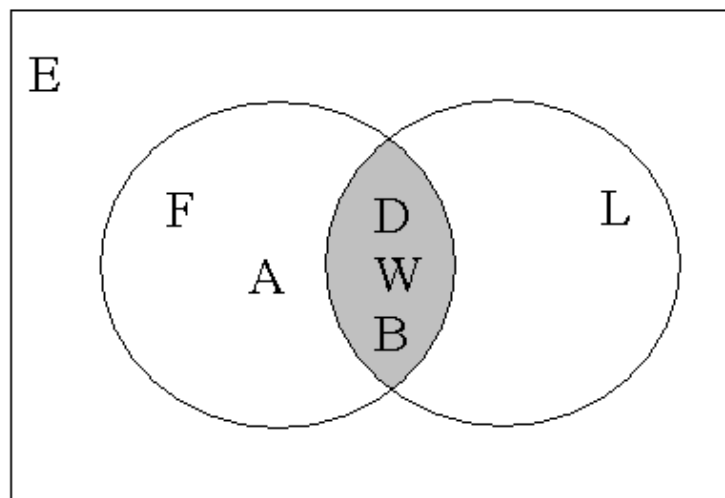
Gr: Employees making more than 35000.



Is this true? [Yes. We see this by examining the diagram.]

Consider (6) again, in terms of sets:

- E: Employees
- F: Females employees
- L: Employees making less than 55000



This shows the counterexample, Anya (A), clearly.

If... then...

Consider:

If an employee is male, then he makes less than 55000.

This is called an “**implication**”. Being a male employee implies making less than 55000. This is a universal quantification in disguise.

The phrase “if... then...” is tricky in English. For example, the meaning of:

- If it rains then there are clouds.
- If there are clouds then it rains.

is clear, but what about:

- You can have dessert if you eat your vegetables.

In English, “if ... then ...” is often intended to mean “if and only if”.

“If” is used loosely in everyday language and it is heavily context dependent. Philosophers, linguists, and logicians explore various meanings of “if” (e.g., cause-and-effect). But we need precision, so we fix the interpretation so that “if P, then Q” is equivalent to “all Ps are Qs”.