Mathematics for Computer Science

José Proença & Alexandre Madeira
(slides from Luis Soares Barbosa)

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY
INESCTEC

Universidade do Minho

Modelling and Calculi
September-October, 2017

Why Maths for Software Engineering?

The phrase **software engineering** dates back to the Garmisch NATO conference in 1968:

> *The phrase software engineering was deliberately chosen as being* **provocative***, in implying the need for software manufacture to be based on the types of* **theoretical foundations** *and practical disciplines, that are* **traditional in the established branches of engineering***.*
>
> (Naur and Randell, 1969)

Provocative or not, 45 years later,
how 'scientific' do such foundations turn out to be?

Science vs Engineering

from *How Science Was Born in 300BC and Why It Had to Be Reborn*:

> *The immense usefulness of exact science consists in providing* **models** *of the real world within which there is a guaranteed method for telling false statements from true. (...) Such models, of course, allow one to* **describe and predict** *natural phenomena, by translating them to the theoretical level via* **correspondence rules**, *then solving the* **exercises** *thus obtained and translating the solutions obtained back to the real world.*

(Lucio Russo, 2007)

Science vs Engineering

- *… providing* ***models*** *of the real world within which there is a guaranteed method for telling false statements from true …*
- *… solving the* ***exercises*** *thus obtained …*

In other words:

- abstract and precise **models** …
- … and calculi for **reasoning about and within** such models

i.e.,

$$\boxed{\text{the hallmark of any } \textbf{true Engineering discipline}}$$

... on second thoughts

IT became ubiquitous to modern life long before a **solid scientific methodology**, let alone formal foundations, has been put forward

Formal modelling aims at lifting to Informatics the **problem-solving strategy** one got used to from school physics:

- understand the problem
- build a mathematical model of it
- animate and reason in such a model
- upgrade the model whenever necessary
- calculate a solution and implement it

... on second thoughts

IT became ubiquitous to modern life long before a **solid scientific methodology**, let alone formal foundations, has been put forward

Formal modelling aims at lifting to Informatics the **problem-solving strategy** one got used to from school physics:

- understand the problem
- build a mathematical model of it
- animate and reason in such a model
- upgrade the model whenever necessary
- calculate a solution and implement it

Models

- introduce **rigour** (in which other engineering would a warranty be replaced by a disclaimer?)
- develop **abstraction** skills
- help developers to **manage complexity** and assess alternative designs in early stages

A good model guides your thinking, a bad one wraps it
Brian Marick

Models

Example

Mary has twice as many apples as John. Mary throws half her apples away, because they are rotten, and John eats one of his. Mary still has twice as many apples as John. How many apples did Mary and John have initially?

Models

Example

$$\begin{cases} m = 2 \cdot j \\ m/2 = 2 \cdot (j - 1) \end{cases}$$

Models

- A mathematical model may be more understandable, concise, precise, or rigorous than an informal description written in a natural language.
- Answers to questions about an object or phenomenon can often be computed directly using a mathematical model of the object or phenomenon.
- Mathematics provides methods for reasoning: for manipulating expressions, for proving properties from and about expressions, and for obtaining new results from known ones. This reasoning can be done without knowing or caring what the symbols being manipulated mean.

Reasoning

Aims at

- calculating solutions (implementations)
- producing evidence (proof, i.e. precise, honest explanations)

$\boxed{\text{Logic is the glue that binds together methods of reasoning}}$

Reasoning

- IT-driven societies require from people a higher degree of mathematical literacy, i.e.
  - the ability to reason in terms of abstract models
  - and effectively use of logical arguments and mathematical calculation
  - ... in daily business practice
- High-valued programmers are heavy users of logic: but a heavy use of logic also requires more concise ways of expression and notations amenable to formal, systematic manipulation

Proofs

Ubiquitous

- certification of critical systems
- e-commerce security
- automatic contracts in global infra-structures (cf, proof-carrying code)

However

- proofs are usually omitted from schoolroom practice
- when present, they are non-systematic, informal, and seen as difficult

What we see

- IT-driven societies require from people a higher degree of mathematical literacy, i.e.
    - the ability to reason in terms of abstract models
    - and effectively use of logical arguments and mathematical calculation
    - ... in daily business practice
- High-valued programmers are heavy users of logic: but a heavy use of logic also requires more concise ways of expression and notations amenable to formal, systematic manipulation

Calculational proofs

Example: is $\log(2) + \log(7) < \log(3) + \log(5)$?

The classical proof: verify the hypothesis

(1) *function* $\log$ *is strictly increasing*

(2) $\log(x \times y) = \log(x) + \log(y)$

(3) $14 < 15$

(4) $14 = 2 \times 7$ *and* $15 = 3 \times 5$

(5) $\log(14) < \log(15)$ *by (1), (3)*

(6) $\log(2) + \log(7) < \log(3) + \log(5)$ *by (2), (4), (5)*

Calculational proofs

The classical proof:

- easy to follow
- but provides poor intuition on the problem
- hard memorize or reproduce
- most probably it was not made, originally, by the order in which it is presented

Calculational proofs

The calculational proof:

$$\begin{array}{ll}
 & \log(2) + \log(7) \;\; \Box \;\; \log(3) + \log(5) \\
= & \quad \{ \;\; \text{function log distributes over multiplication} \;\; \} \\
 & \log(2 \times 7) \;\; \Box \;\; \log(3 \times 5) \\
= & \quad \{ \;\; \text{routine arithmetic} \;\; \} \\
 & \log(14) \;\; \Box \;\; \log(15) \\
= & \quad \{ \;\; 14 < 15 \text{ and function log is strictly increasing} \;\; \} \\
 & \;\Box \quad \text{is} \quad <
\end{array}$$

Calculational proofs

The calculational proof:

- The starting point is not an hypothesis to verify, but the problem itself.
- The proof starts by identifying an unknown □ which stands for an order relation
- and proceeds by the identification and application of whatever known properties are useful in its determination
- being essentially syntax driven it builds intuition and meaning.

Syllabus

- Logic: propositional and first order
- Set theory
- Functions and relations
- Partial orders, monoids, trees and graphs
- Recurrence relations and induction

Pragmatics

18 Sep – 28 Sep (4 weeks)

- 18 Sep – 28 Sep (2 weeks), José Proença
- 2 Oct – 12 Oct (2 weeks), Alexandre Madeira

Test: 10 and 12 October
Exam: 13 to 20 December

References

- A logical approach to discrete mathematics, David Gries and Fred Schneider, Springer, 1993.
- Discrete mathematics, Kevin Ferland, Houghton Mifflin Company, 2009.
- ... plus slides and exercises ...

Other references

- Naive Set Theory, Paul Halmos, 1960

`http://mi1617.proenca.org`