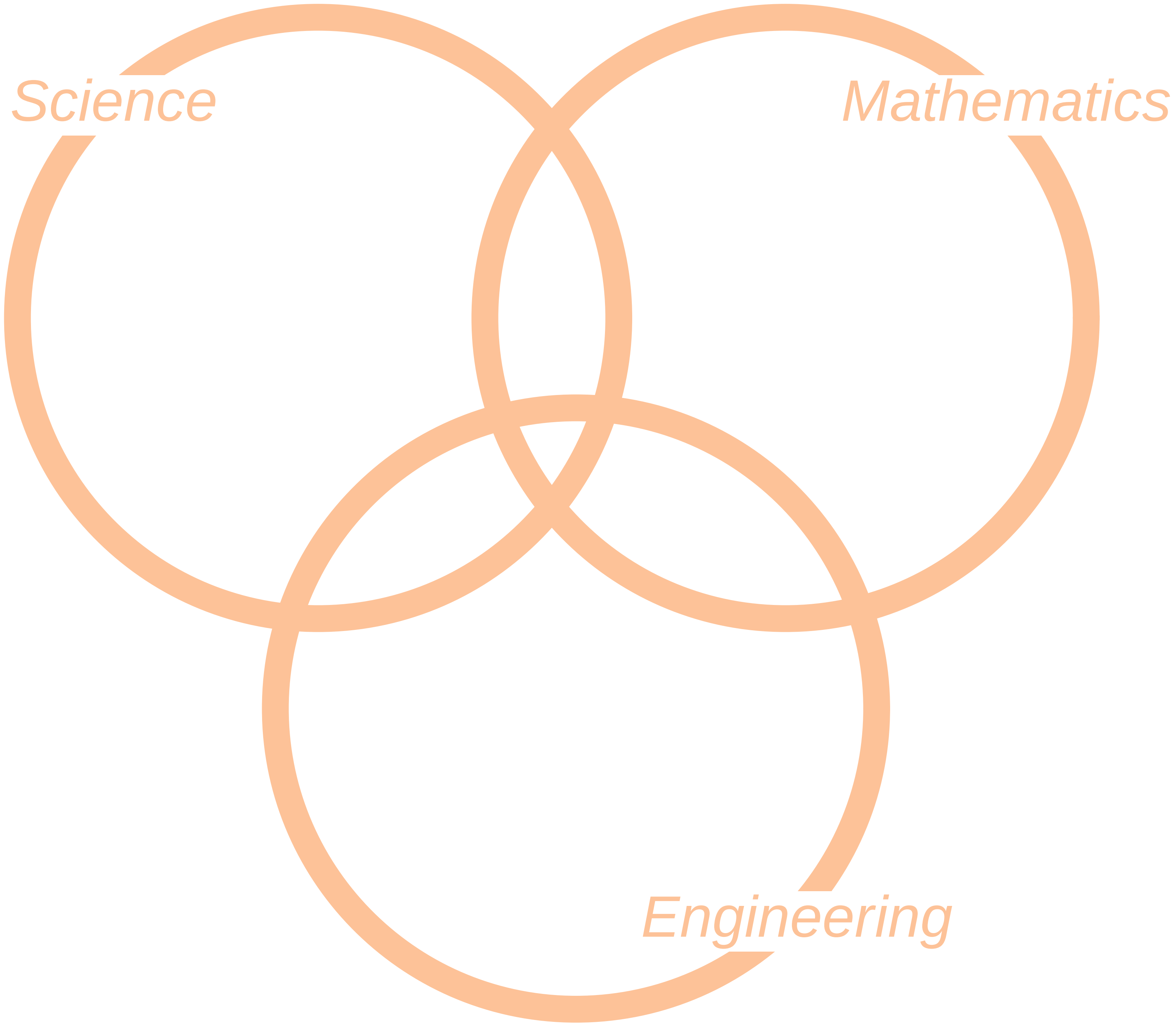
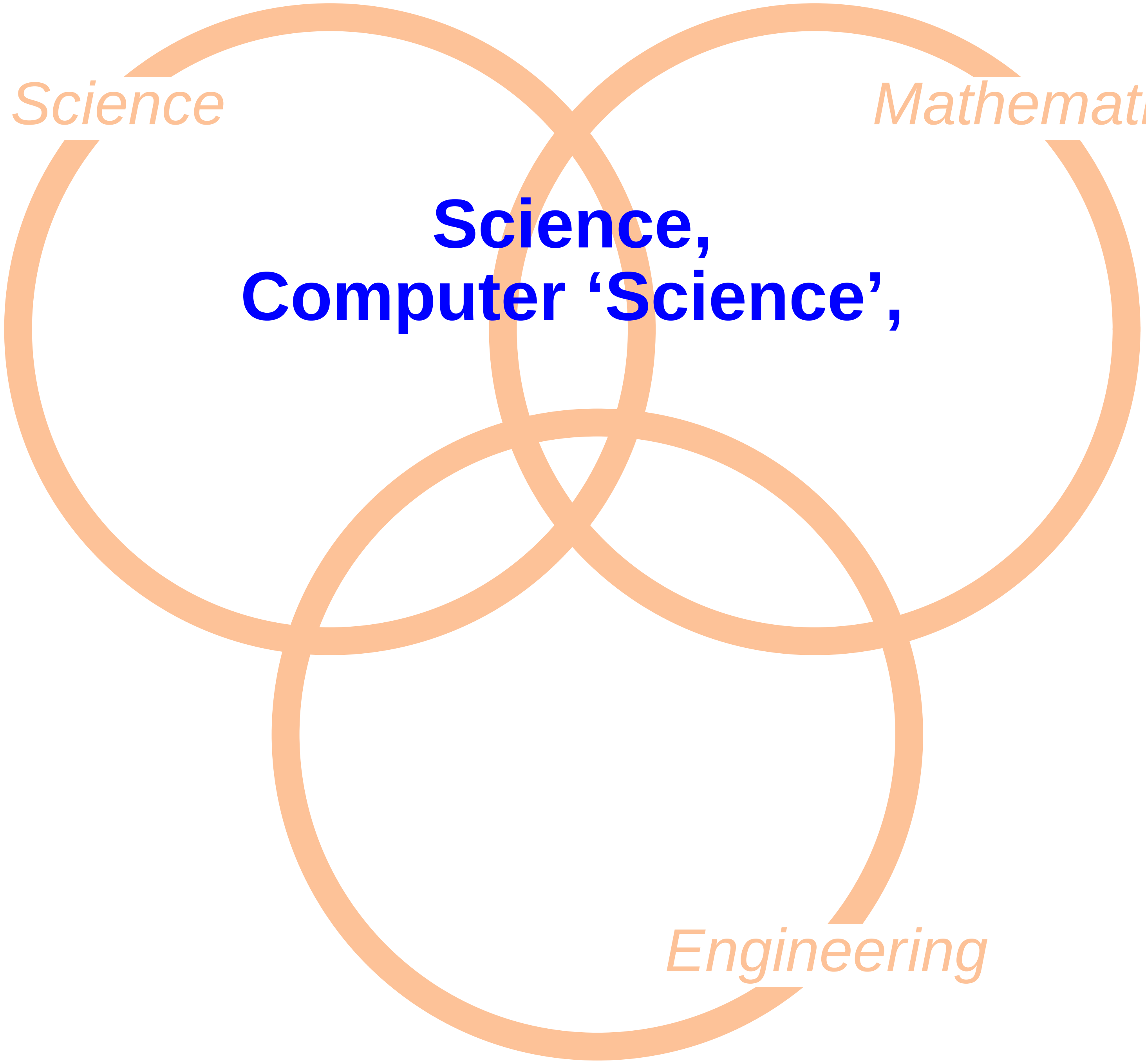
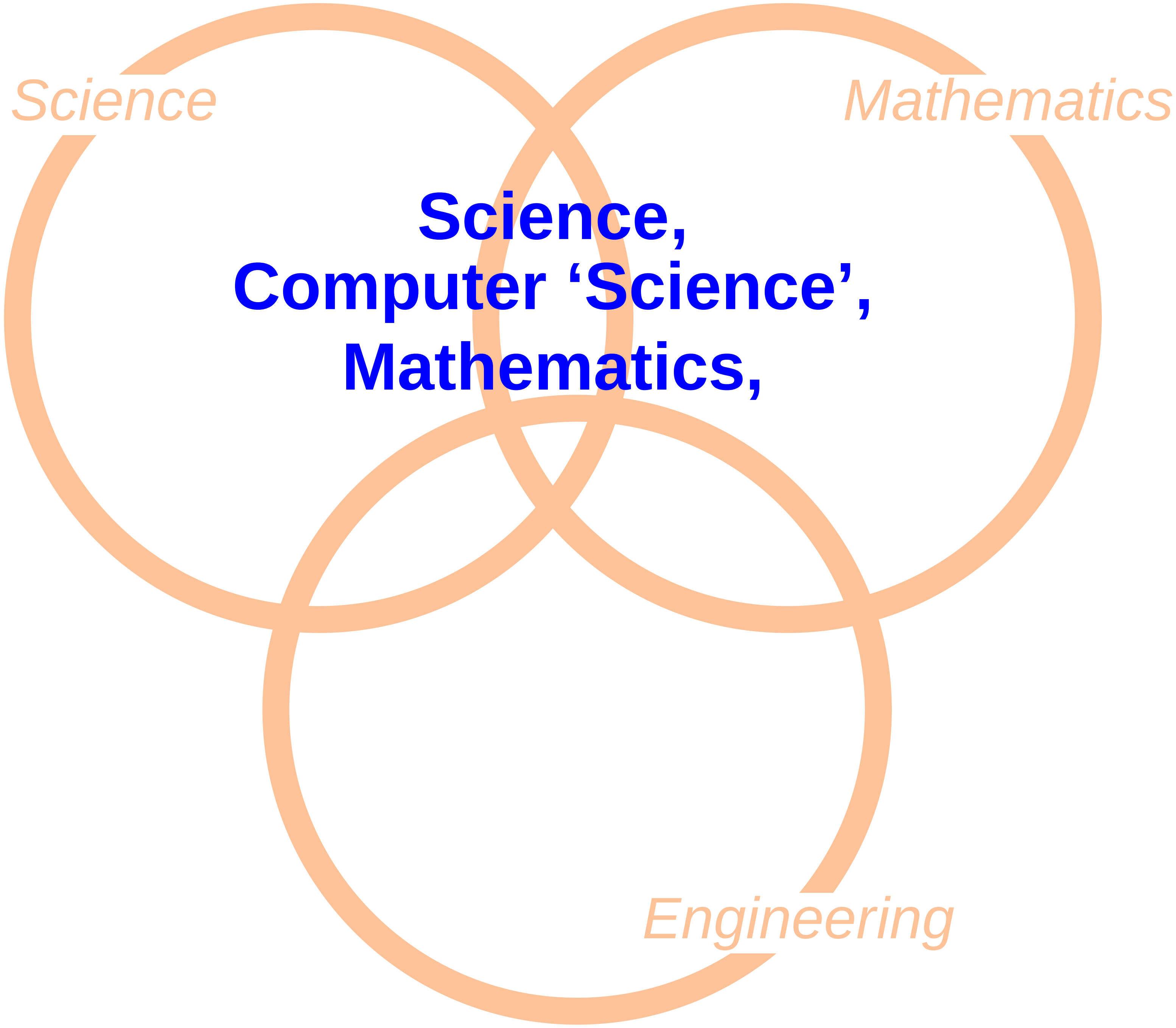
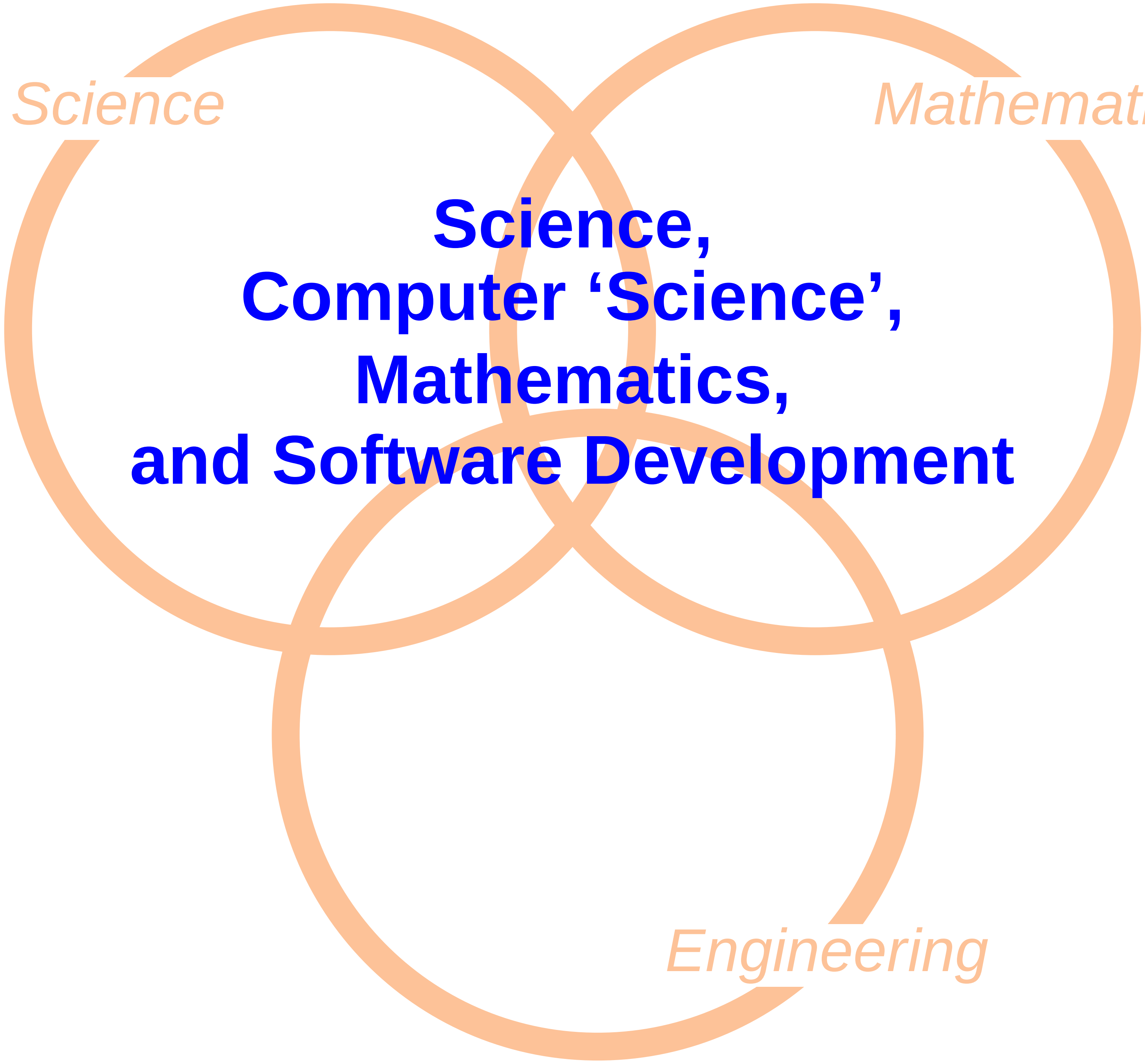

Science
Mathematics
Engineering



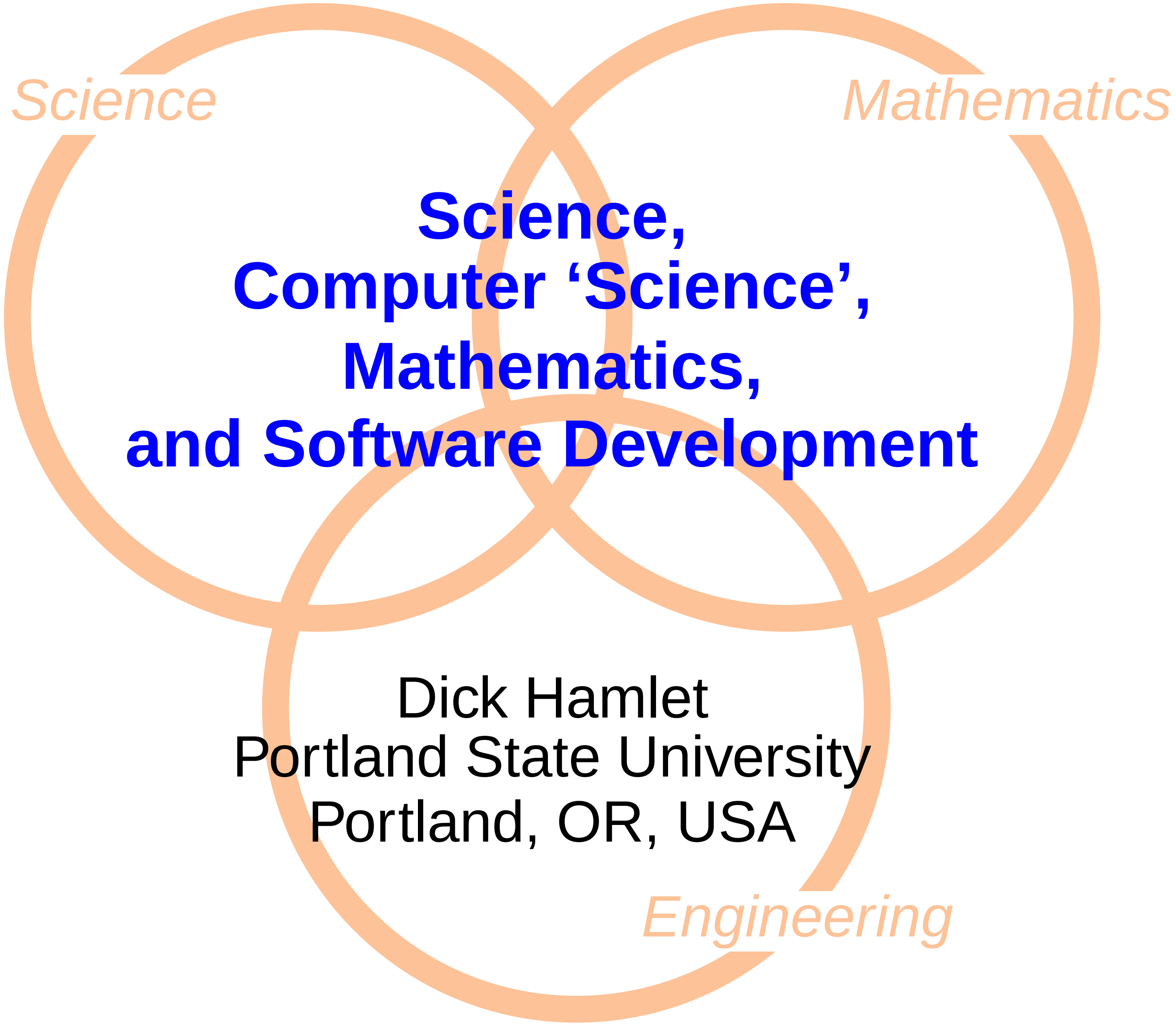
Science,
Computer 'Science',



Science,
Computer ‘Science’,
Mathematics,



Science,
Computer ‘Science’,
Mathematics,
and Software Development



Science, Computer ‘Science’, Mathematics, and Software Development

Dick Hamlet
Portland State University
Portland, OR, USA

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

Fundamental Questions

- $\boxed{\text{What is software engineering?}}$

“Whatever it is I do all day...”

... is not much good

Fundamental Questions

- $\boxed{\text{What is software engineering?}}$

“Whatever it is I do all day...”

... is not much good

There are good answers to similar questions:

- What is physics?
- What is civil engineering?

(Some of those answers along the way)

What Use is Philosophy?

- Make small talk at parties...
- Feel good (or bad) about your profession...

What Use is Philosophy?

- Make small talk at parties...
- Feel good (or bad) about your profession...
- Philosophy can influence your work...
  - Newton was seeking God's truth

What Use is Philosophy?

- ▶ Make small talk at parties...
- ▶ Feel good (or bad) about your profession...
- ▶ Philosophy can influence your work...
  - ▷ Newton was seeking God’s truth

“We all have our philosophies...and [they] are not worth very much. But [their] impact upon our actions and our lives is often devastating.”
–Karl Popper

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

Outline of the Talk

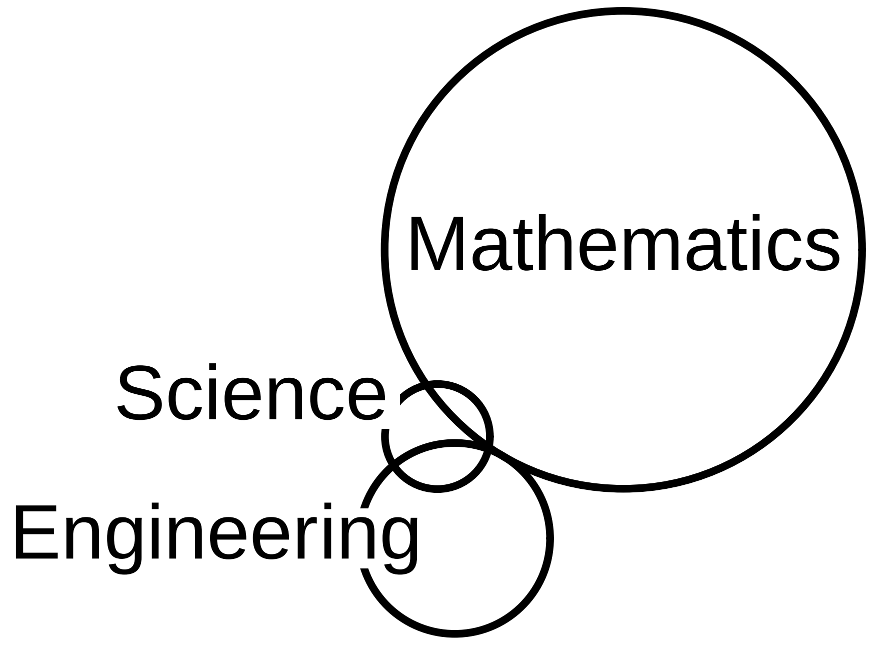
I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

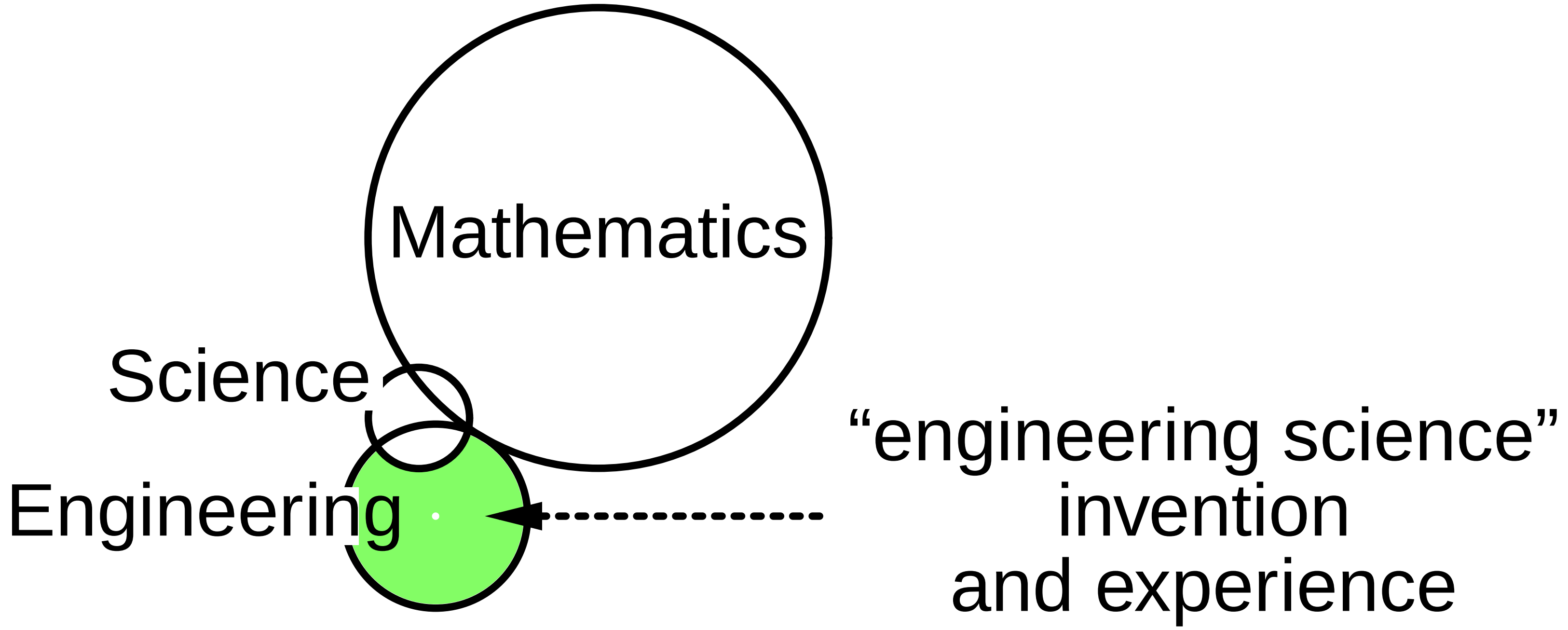
IV. What's To Be Done About It?

Science, Math, and Engineering

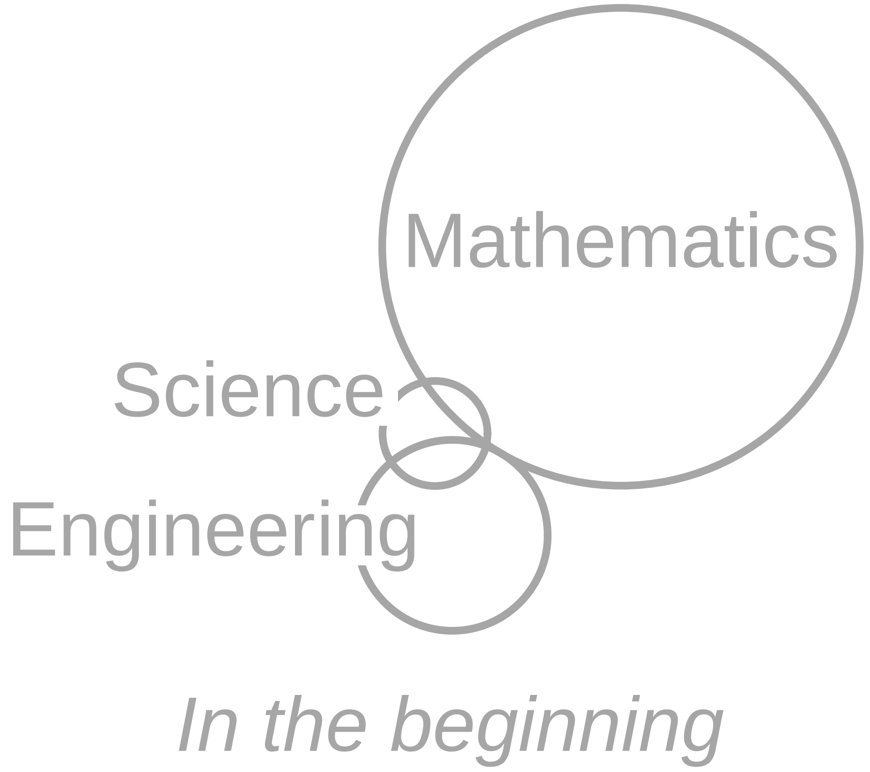


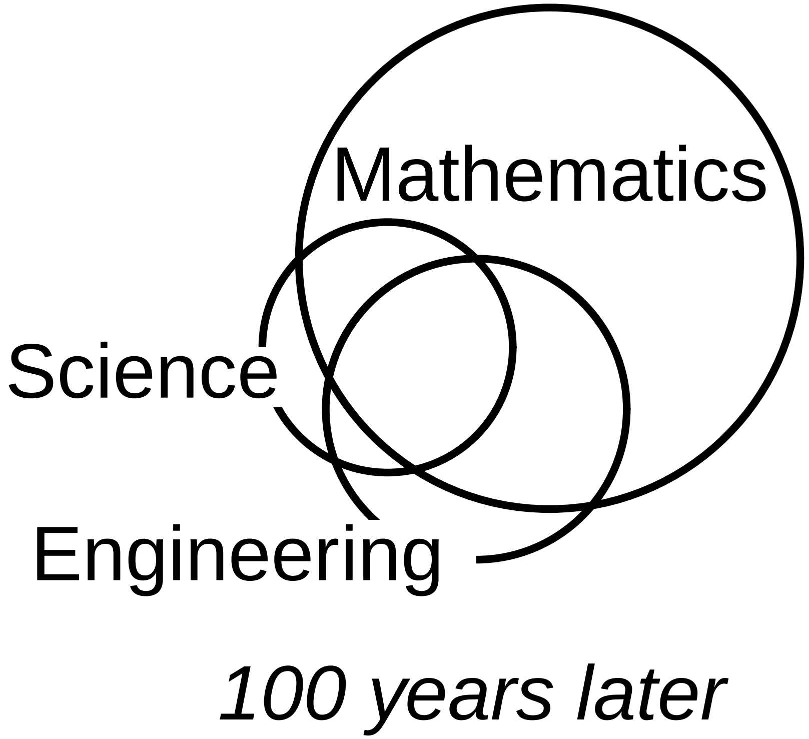
In the beginning

Science, Math, and Engineering



Science, Math, and Engineering





What Are The Disciplines?

Questions answered by:

Mathematics:

How to describe complicated things?

Science:

How does the world really work?
What are its natural laws?

Engineering:

How can I make it happen as I want?

What Are The Disciplines?

Questions answered by:

Mathematics:

How to describe complicated things?

Science:

How does the world really work?
What are its natural laws?

Engineering:

How can I make it happen as I want?

Science & Mathematics	*Technology*
Understanding of reality	Human **control**

Formal Definitions (c. 2000)

(Mathematical) theory: A body of definitions, axioms, and theorems.

(Scientific) theory: A collection of assertions about reality that may be falsified by observation.
Ideally, applied mathematics.

Engineering design rules (Theory?): Systematic procedures for making artifacts, drawn from science, practice, invention, and experiment.

Applied Mathematics

- Start with a rich mathematical theory
- Identify the theory’s objects with physical entities
- Check that the theory’s axioms are true for those entities
- Exploit the theorems of the theory

Applied Mathematics

- Start with a rich mathematical theory
- Identify the theory's objects with physical entities
- Check that the theory's axioms are true for those entities
- Exploit the theorems of the theory

☞ Creates a precise scientific theory ☜

Testing a Scientific Theory

Suppose a theorem of the mathematical theory (in the scientific theory) is observed not to hold.

We say, “The theory is wrong,” meaning the scientific one.

- ▶ The axioms did *not* hold.

  OR

- ▶ Mathematical logic is wrong.

Testing a Scientific Theory

Suppose a theorem of the mathematical theory (in the scientific theory) is observed not to hold.

We say, “The theory is wrong,” meaning the scientific one.

- $\boxed{\text{The axioms did } \textit{not} \text{ hold.}}$

  OR

- Mathematical logic is wrong.

We much prefer this one

‘Normal’ Engineering Design

- Design rules tried and true, used before
- No new ‘engineering science’ allowed
- ‘Safety factors’ cover design errors
- Very likely to succeed in use
- Rare failures publicly analyzed

‘Normal’ Engineering Design

- Design rules tried and true, used before
- No new ‘engineering science’ allowed
- ‘Safety factors’ cover design errors
- Very likely to succeed in use
- Rare failures publicly analyzed

In ‘pre-normal’ times engineers can’t work properly

Many failures or innovations force design-rule change – ending a ‘normal’ period

(See Addis, inspired by Kuhn)

Summary: Traditional Paradigm

Science seeks to accurately describe the world's laws

Engineering design must conform to scientific laws – normal design removes some of the uncertainty (See Vincenti)

Mathematics is the handmaiden of science, the tool of engineering

Summary: Traditional Paradigm

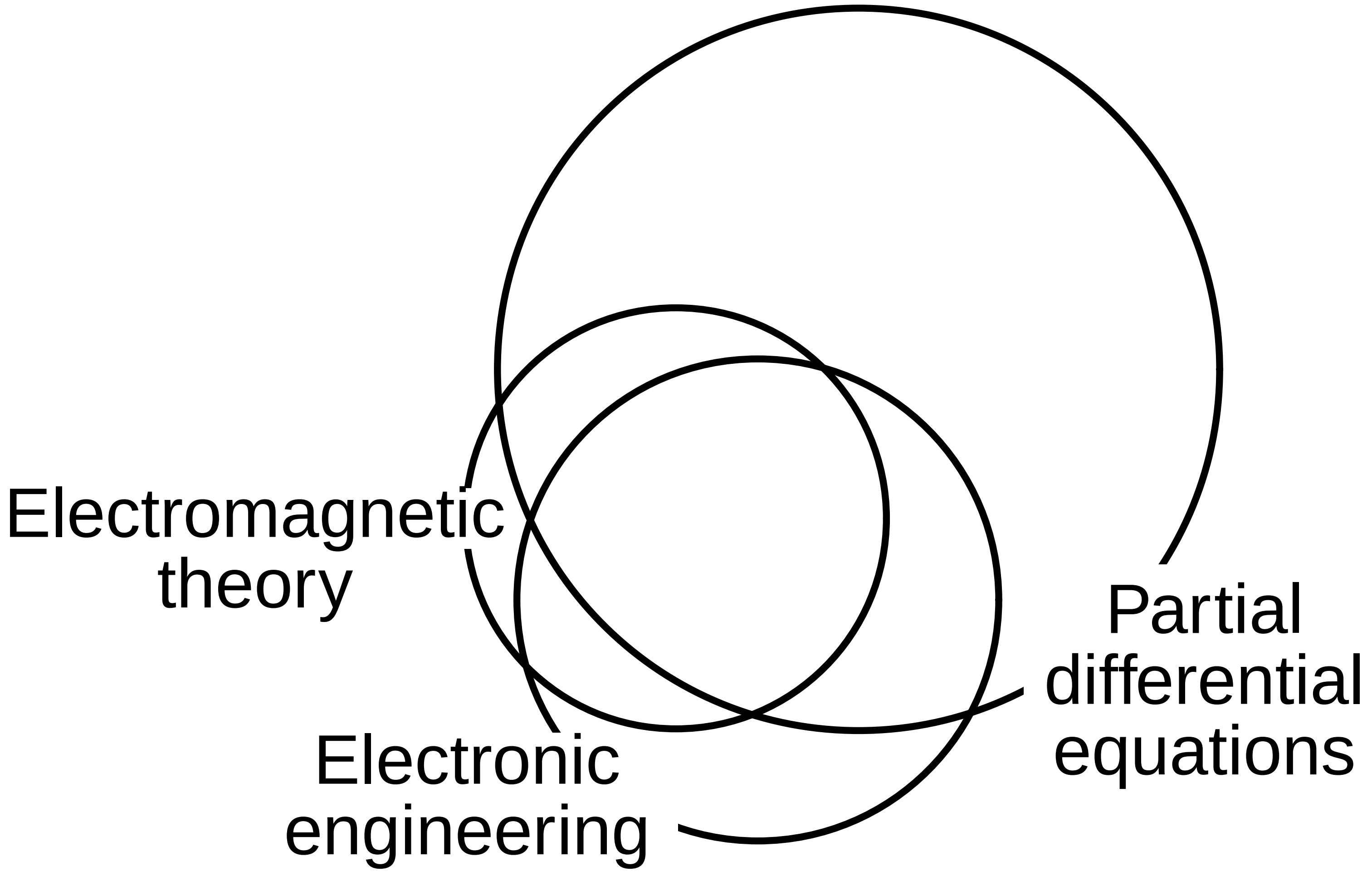
Science seeks to accurately describe the world's laws

Engineering design must conform to scientific laws – normal design removes some of the uncertainty (See Vincenti)

Mathematics is the handmaiden of science, the tool of engineering

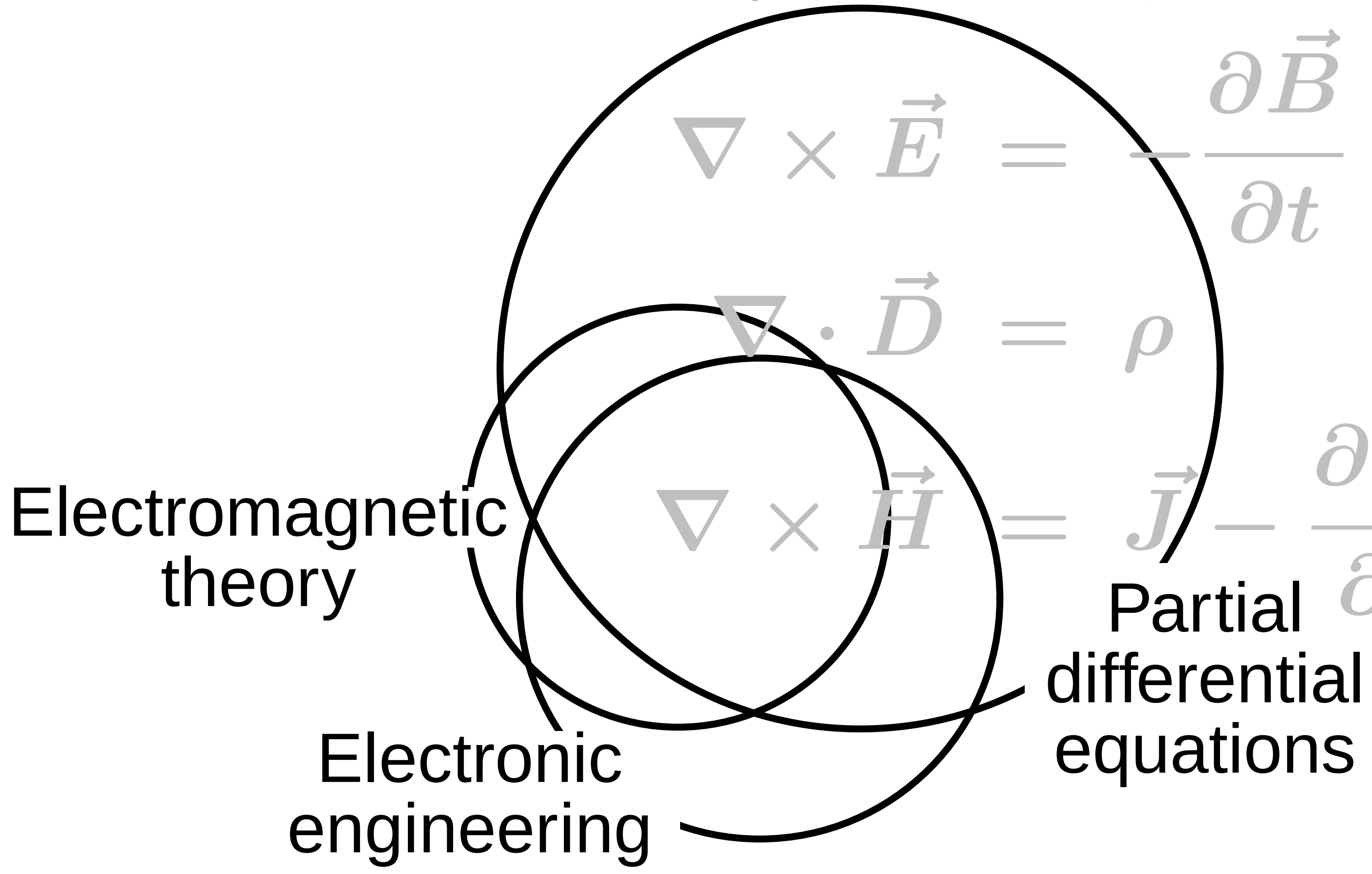
☞ Spectacular success in mechanical, civil, aeronautical, and electrical engineering ☜

Traditional Engineering (Electrical)



Traditional Engineering (Electrical)

$$\begin{aligned}
\nabla \cdot \vec{B} &= 0 \\
\nabla \times \vec{E} &= -\frac{\partial \vec{B}}{\partial t} \\
\nabla \cdot \vec{D} &= \rho \\
\nabla \times \vec{H} &= \vec{J} - \frac{\partial \vec{D}}{\partial t}
\end{aligned}$$



☞ Maxwell equations: applied mathematics of electromagnetic theory ☜

Today’s Philosophical Truths

- **Mathematics** isn’t true or false.
  Mathematical objects are merely “the things that satisfy the axioms” (if any)
  We hope they also satisfy the theorems

Today’s Philosophical Truths

- **Mathematics** isn’t true or false.
  Mathematical objects are merely “the things that satisfy the axioms” (if any)
  We hope they also satisfy the theorems
- **Science** isn’t objective, but ‘theory saturated’.
  Science starts with a problem to be explained, then comes a theory, and finally observations testing theory (Karl Popper)

Today’s Philosophical Truths

- **Mathematics** isn’t true or false.
  Mathematical objects are merely “the things that satisfy the axioms” (if any)
  We hope they also satisfy the theorems
- **Science** isn’t objective, but ‘theory saturated’.
  Science starts with a problem to be explained, then comes a theory, and finally observations testing theory (Karl Popper)
- **Engineering** design rules must be usable – they don’t have to be scientific.
  Safety factors compensate for incorrect theories in the rules

Quotations Supporting the Truths

- “Mathematics is the subject in which we never know what we are talking about, nor whether what we are saying is true.” –Bertrand Russell
- “It is also a good rule not to put overmuch confidence in the observational results that are put forward until they are confirmed by theory.” –Sir Arthur Eddington
- “The Doric design procedures ... were elegantly simple... They required the selection of a single fundamental ‘module’ equal to one half the diameter of a column, all parts of the work adjusted by means of calculations based upon it.” –William Addis

Quotations Supporting the Truths

- “Mathematics is the subject in which we never know what we are talking about, nor whether what we are saying is true.” –Bertrand Russell
- “It is also a good rule not to put overmuch confidence in the observational results that are put forward until they are confirmed by theory.” –Sir Arthur Eddington
- “The Doric design procedures ... were elegantly simple... They required the selection of a single fundamental ‘module’ equal to one half the diameter of a column, all parts of the work adjusted by means of calculations based upon it.” –William Addis

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

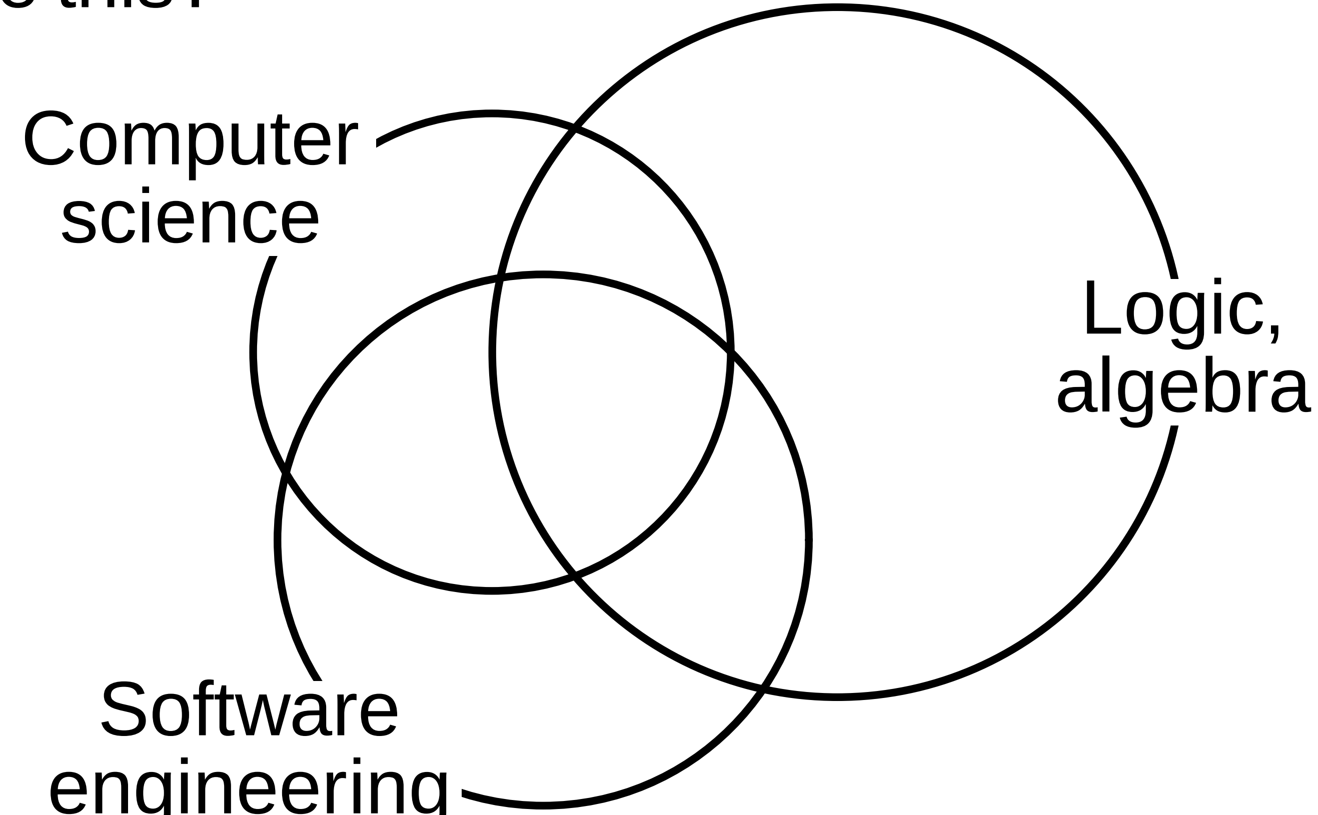
Software *Engineering*?

- It *is* technology – software controls the world today

Software *Engineering*?

- It *is* technology – software controls the world today

Is it like this?



Computer *Science*?

- Donald Knuth's volumes are entitled
  The <u>Art</u> of Computer Programming

Computer *Science*?

- Donald Knuth’s volumes are entitled
  The <u>Art</u> of Computer Programming
- C.A.R. Hoare (q.v.) looks (forward!) to ‘laws’ and ‘science’ for programming. But (on Ada):

Computer *Science*?

- Donald Knuth's volumes are entitled *The Art of Computer Programming*
- C.A.R. Hoare (q.v.) looks (forward!) to 'laws' and 'science' for programming. But (on Ada):

  > "Almost anything in software can be implemented, and even used, given enough determination."

Computer *Science*?

- Donald Knuth's volumes are entitled
  The <u>Art</u> of Computer Programming
- C.A.R. Hoare (q.v.) looks (forward!) to 'laws' and 'science' for programming. But (on Ada):

  > "Almost anything in software can be implemented, and even used, given enough determination."

  - Not even the DoD can buy physical laws

Computer *Science*?

- Donald Knuth's volumes are entitled
  The <u>Art</u> of Computer Programming
- C.A.R. Hoare (q.v.) looks (forward!) to 'laws' and 'science' for programming. But (on Ada):

  > "Almost anything in software can be implemented, and even used, given enough determination."

  - Not even the DoD can buy physical laws
- In an accredited CS curriculum physics and calculus are required but not used

Computer *Science*?

- Donald Knuth's volumes are entitled *The Art of Computer Programming*
- C.A.R. Hoare (q.v.) looks (forward!) to 'laws' and 'science' for programming. But (on Ada):

  "Almost anything in software can be implemented, and even used, given enough determination."

  - Not even the DoD can buy physical laws
- In an accredited CS curriculum physics and calculus are required but not used
- "If a discipline has 'science' in its name, it isn't."

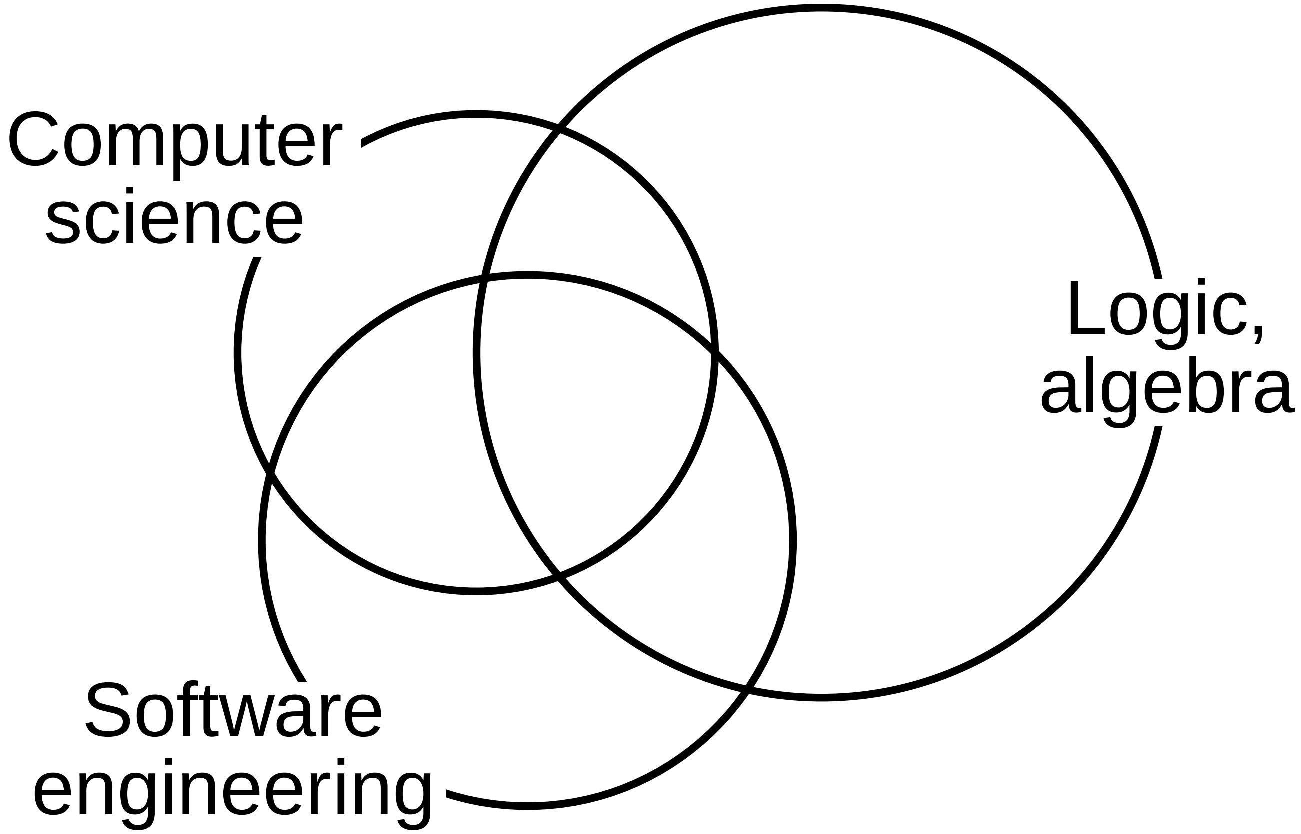
What is Programming?

- Programming is at the heart of CS
- There is no science of programming
  - Programming skill is taught by example
  - Particular programs are studied as artifacts

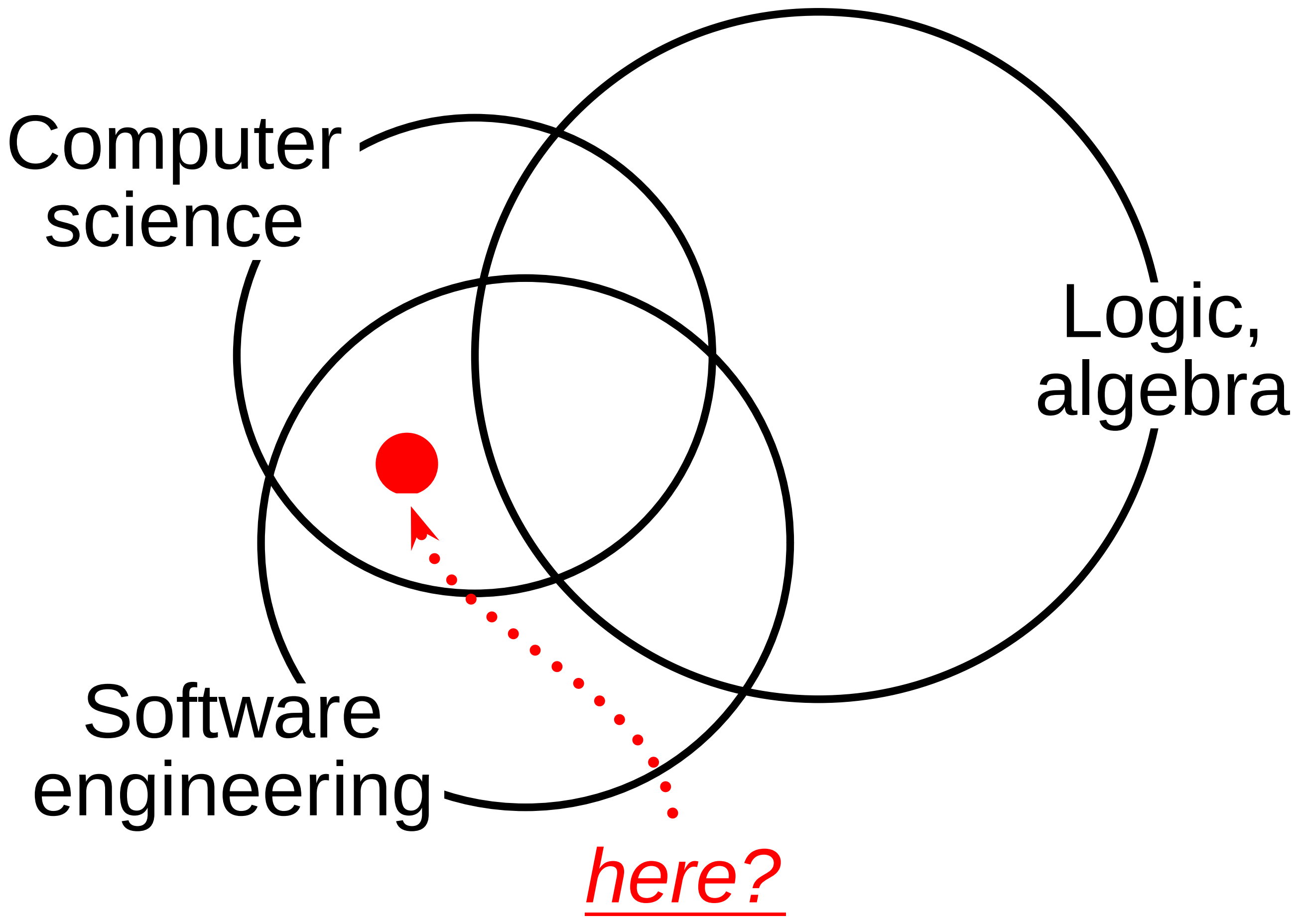
What is Programming?

- Programming is at the heart of CS
- There is no science of programming
  - Programming skill is taught by example
  - Particular programs are studied as artifacts
- What are the laws of programming?
  - You must use C++?
  - You must choose identifiers to company standard?
  - You must not write:
    `if X = Y` for `if (X == Y)` ?

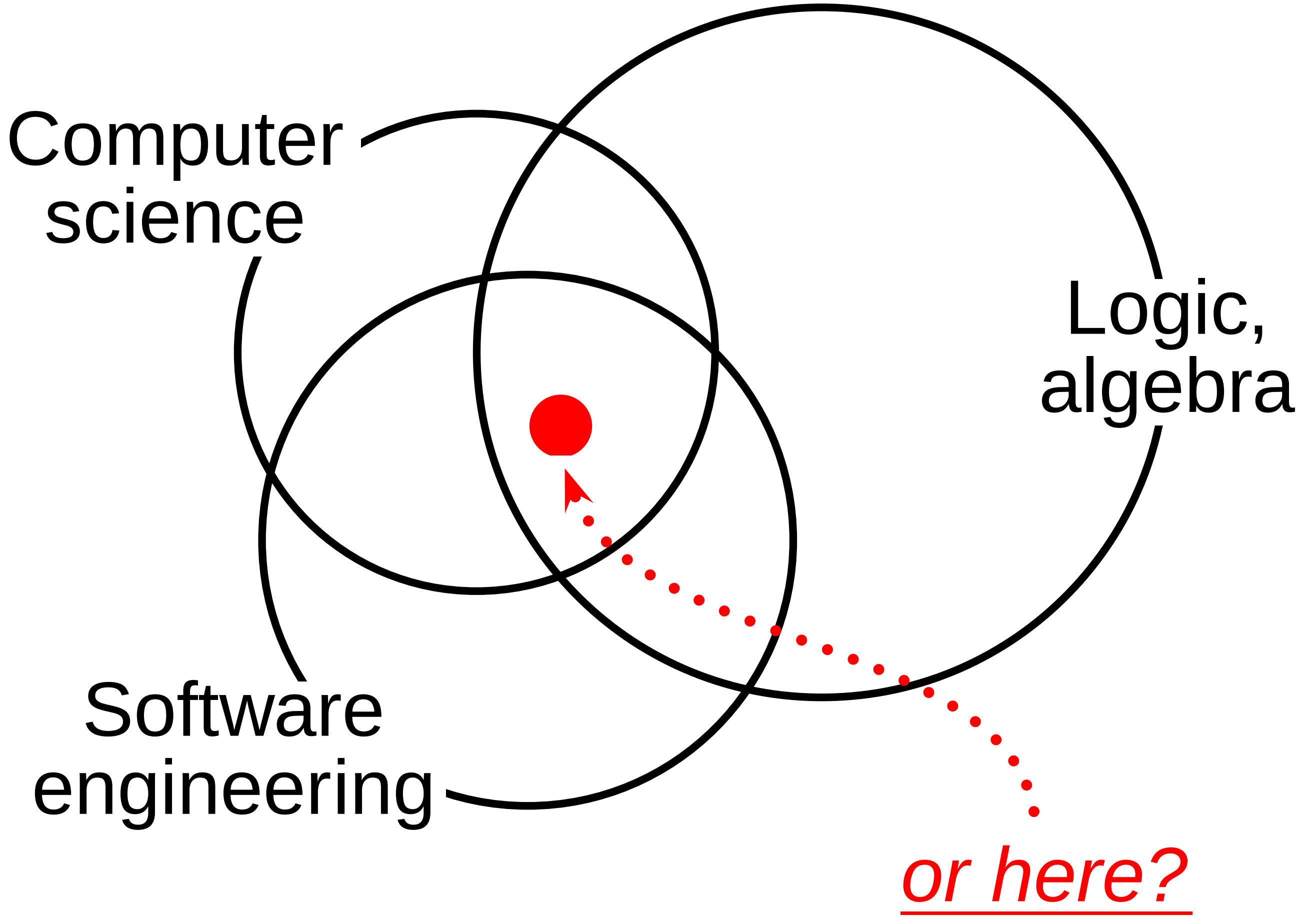
Where Does Programming Fit?



Where Does Programming Fit?



Where Does Programming Fit?



Computer ‘Science’ Isn’t

- There are no falsifying experiments
  - ‘Experiment’ in CS means to implement an idea and force it to work
- Programming languages are invented
  - They can be changed at will (past time!)
  - Language properties can be *proved* (and if the proof fails, *changed*)

Computer ‘Science’ Isn’t

- There are no falsifying experiments
  - ‘Experiment’ in CS means to implement an idea and force it to work
- Programming languages are invented
  - They can be changed at will (past time!)
  - Language properties can be *proved* (and if the proof fails, *changed*)

☞ Scientists can’t change reality to fit theory ☜

Digression: There’s *Some* Science...

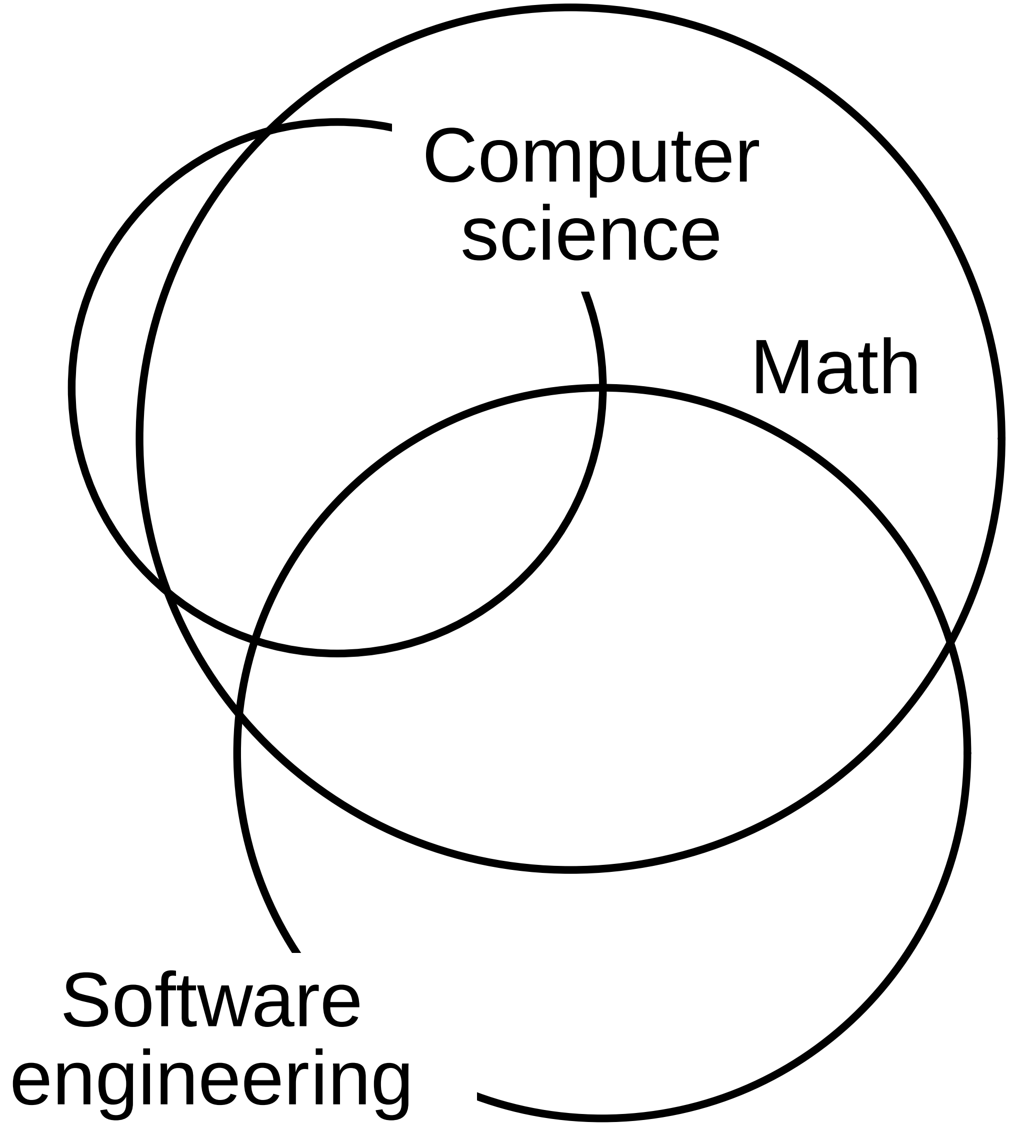
- Information theory and undecidability (complexity) theory are something like thermodynamic laws
- ‘Science’ overlaps with ‘rational discussion’

  “I may be wrong and you may be right, and by an effort we may get nearer to the truth.”

  –Karl Popper

  - Mathematics and CS: consistency, depth, and elegance replace experiments
- The sociology and economics of software engineering are human laws, but they have immense inertia

Computer *Mathematics*?



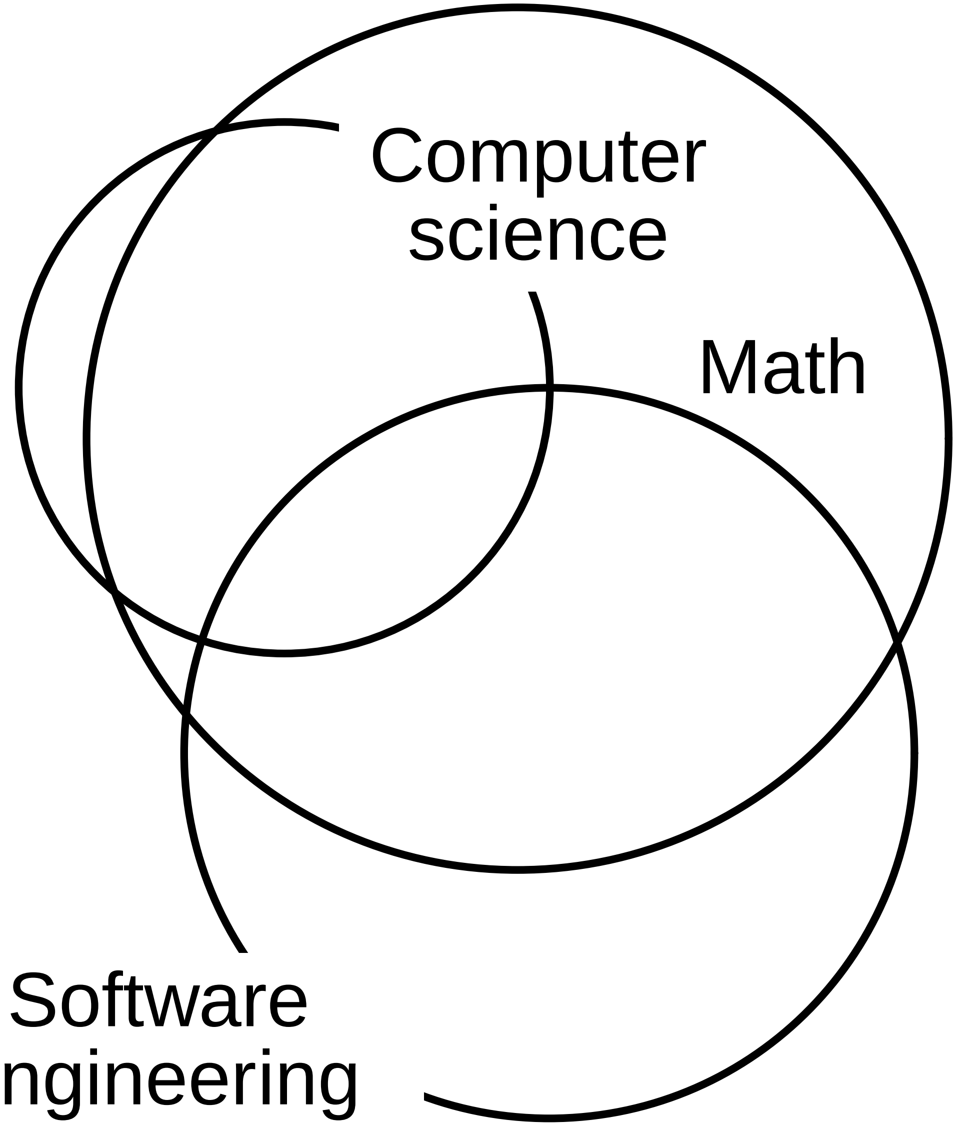
Computer *Mathematics*?

information theory

programming

economics of SE

sociology of SE



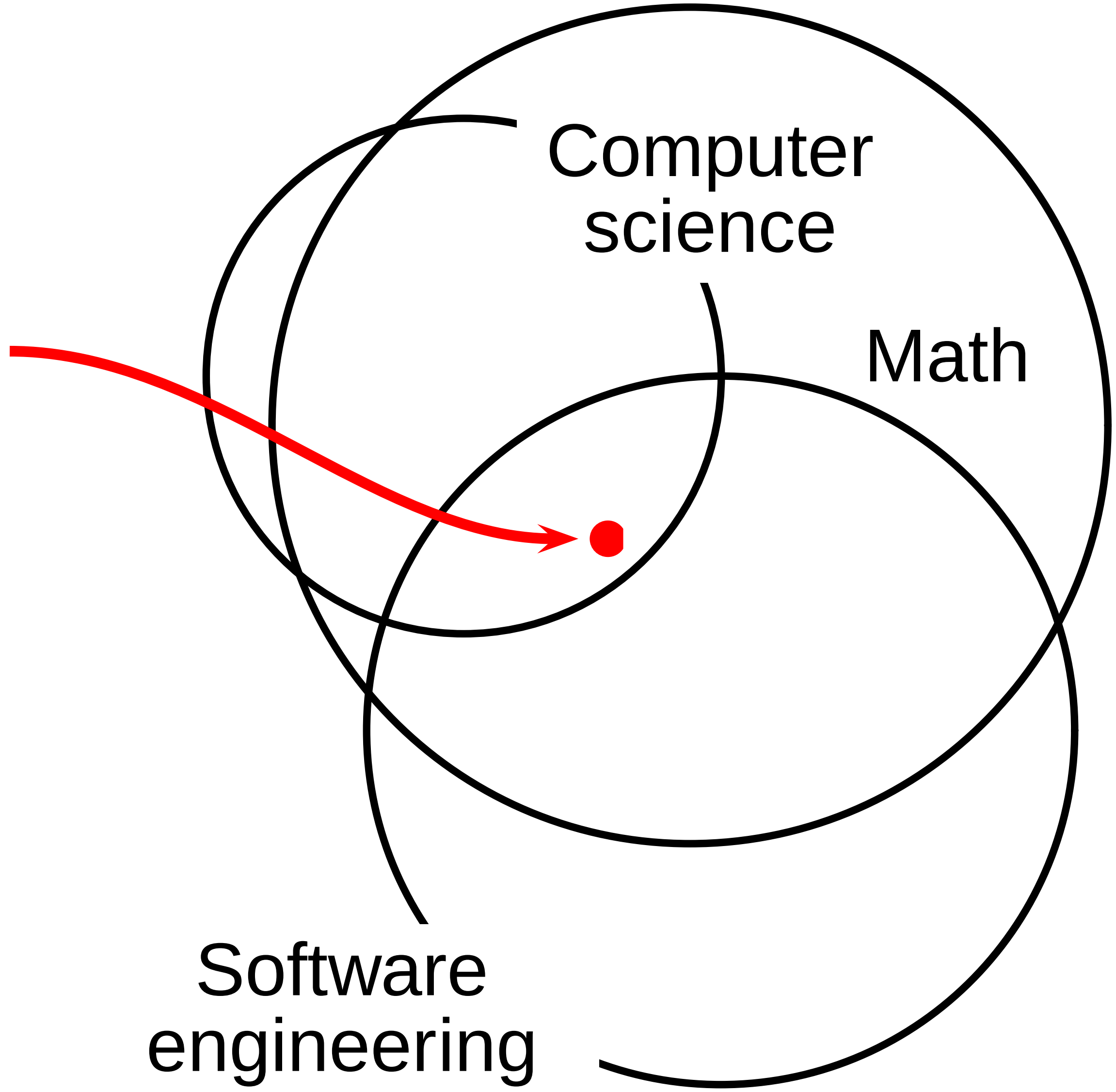
Computer *Mathematics?*

information theory

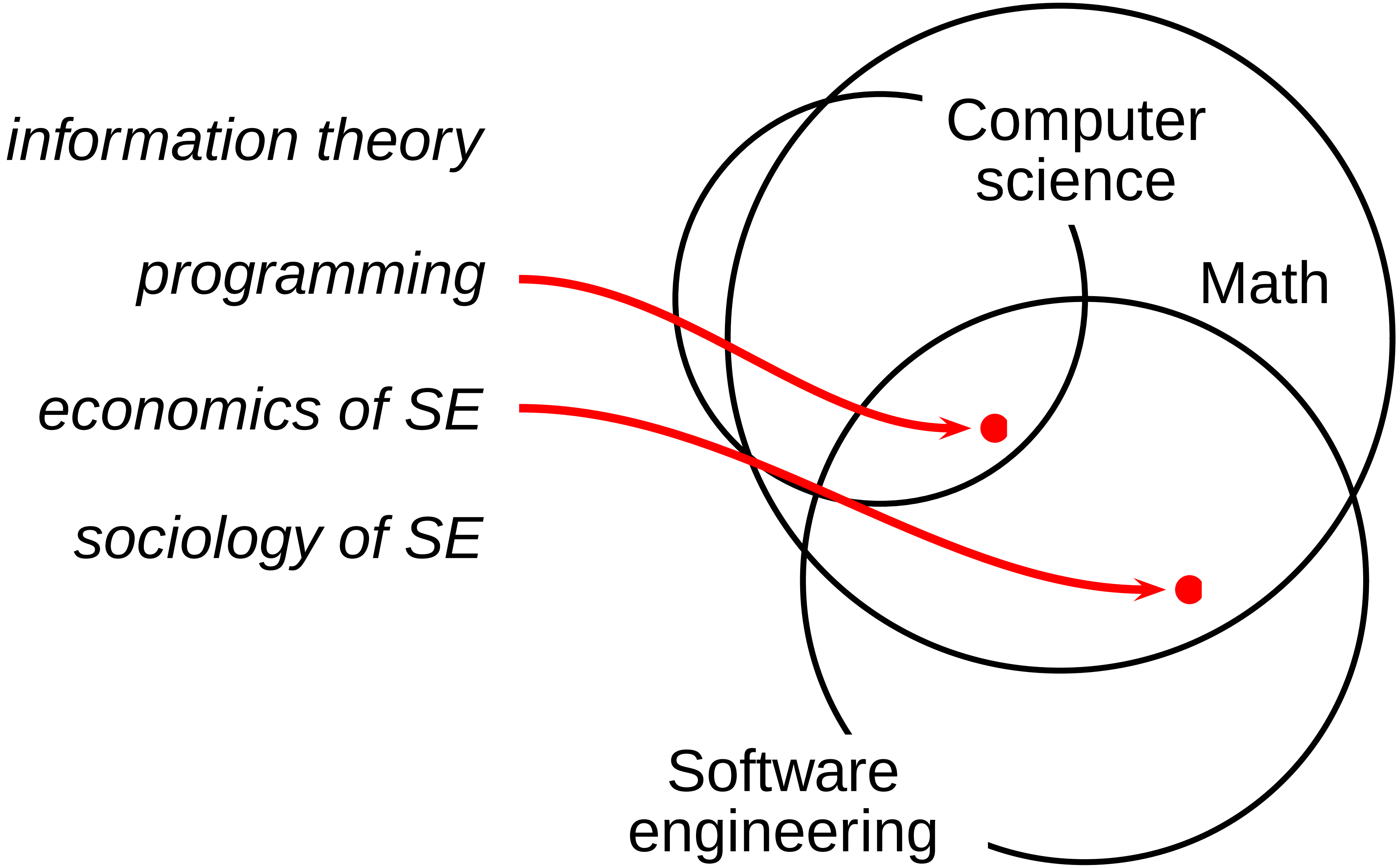
programming

economics of SE

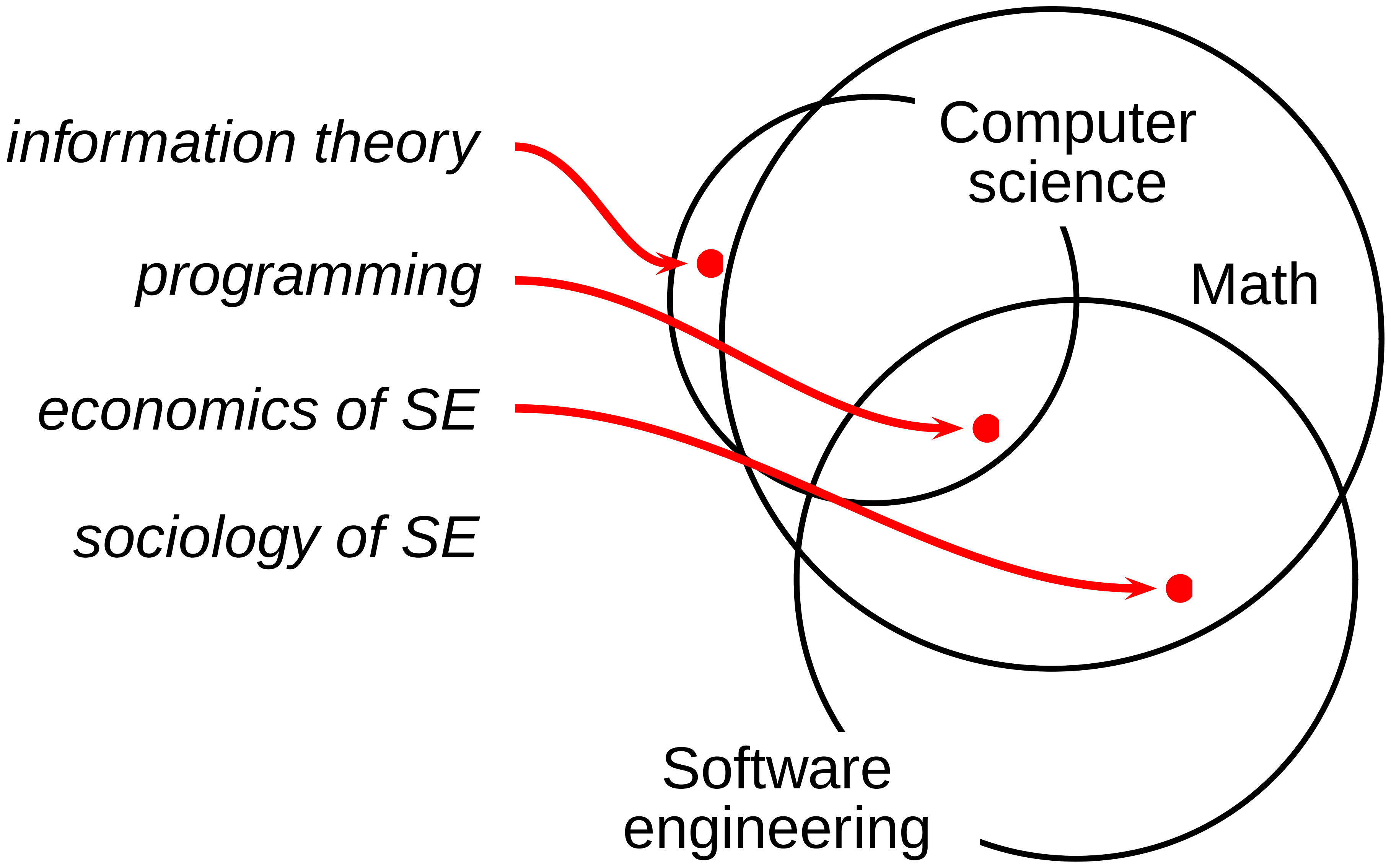
sociology of SE



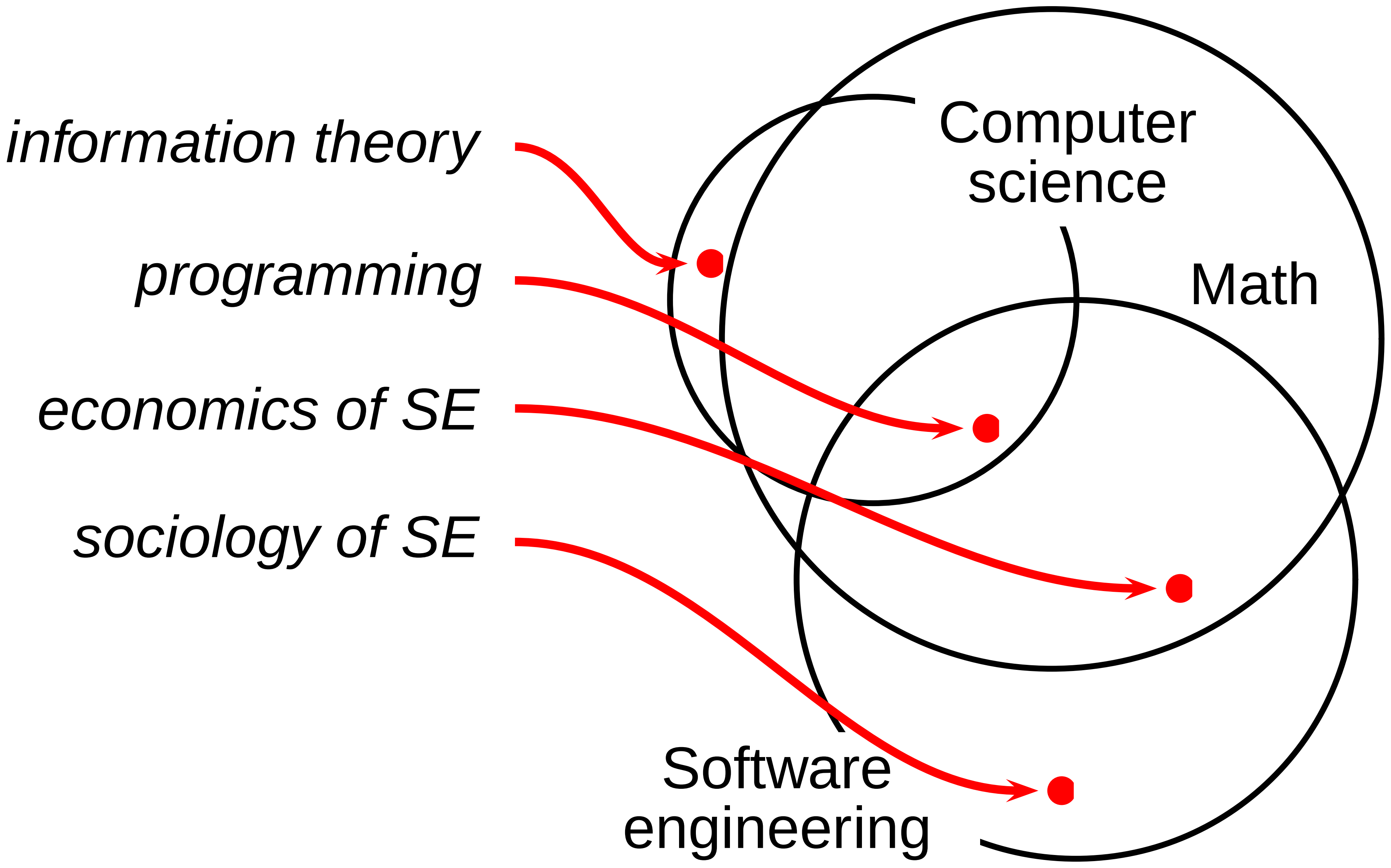
Computer *Mathematics?*



Computer *Mathematics*?



Computer *Mathematics*?



Evaluating Engineering Designs

- Solving problems – an analogy:
  - Civil engineering: Design a bridge consistent with reality (impossible?)
  - Software engineering: Design a program consistent with customer requirements (??)

Evaluating Engineering Designs

- Solving problems – an analogy:
  - Civil engineering: Design a bridge consistent with reality (impossible?)
  - Software engineering: Design a program consistent with customer requirements (??)
- Einstein said that in making the laws of physics, God was “subtle, but not damn mean”
  - Sometimes customers *are* damn mean...

Evaluating Engineering Designs

- Solving problems – an analogy:
  - Civil engineering: Design a bridge consistent with reality (impossible?)
  - Software engineering: Design a program consistent with customer requirements (??)
- Einstein said that in making the laws of physics, God was “subtle, but not damn mean”
  - Sometimes customers *are* damn mean...

☞ How is a good design different from a poor one? ☜

Examples of Models in Engineering

- ▶ Aeronautical engineering: wing behavior
  - ▷ May be proved wrong by experiment
  - ▷ Mistakes covered by a safety factor
- ▶ Software engineering: efficacy of testing
  - ▷ Can be verified by mathematical proof
  - ▷ Mistakes may be arbitrarily bad

☞ In software, a model is mathematics to explain some other mathematics ☜

Software Modeling

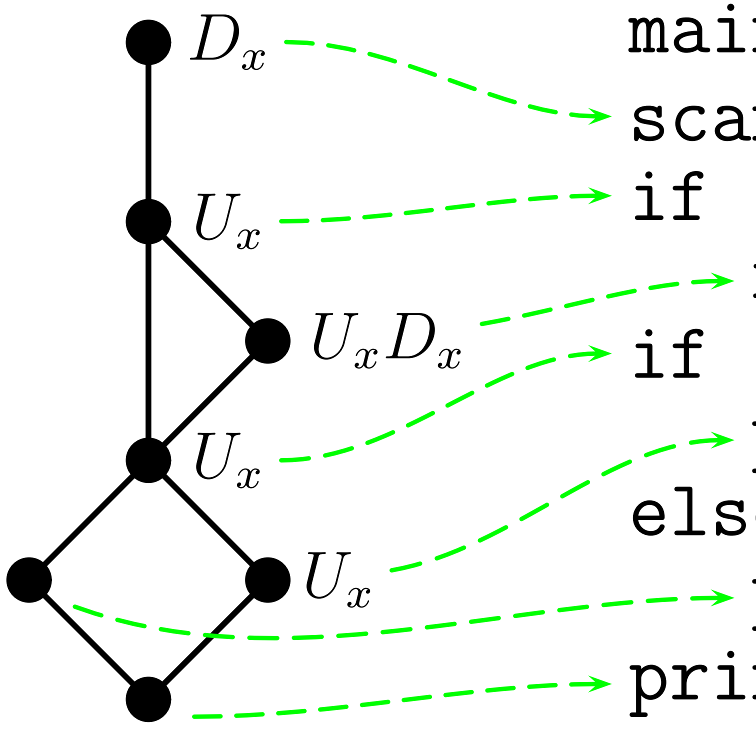
‘Reality’

```
main() { int x;
scanf("%d", &x);
if (x >= 100)
   x += 100;
if (x > 150 && x < 170)
   printf("%d", x+1);
else
   printf("Error");
printf("Done");
}
```

Software Modeling

Model

‘Reality’

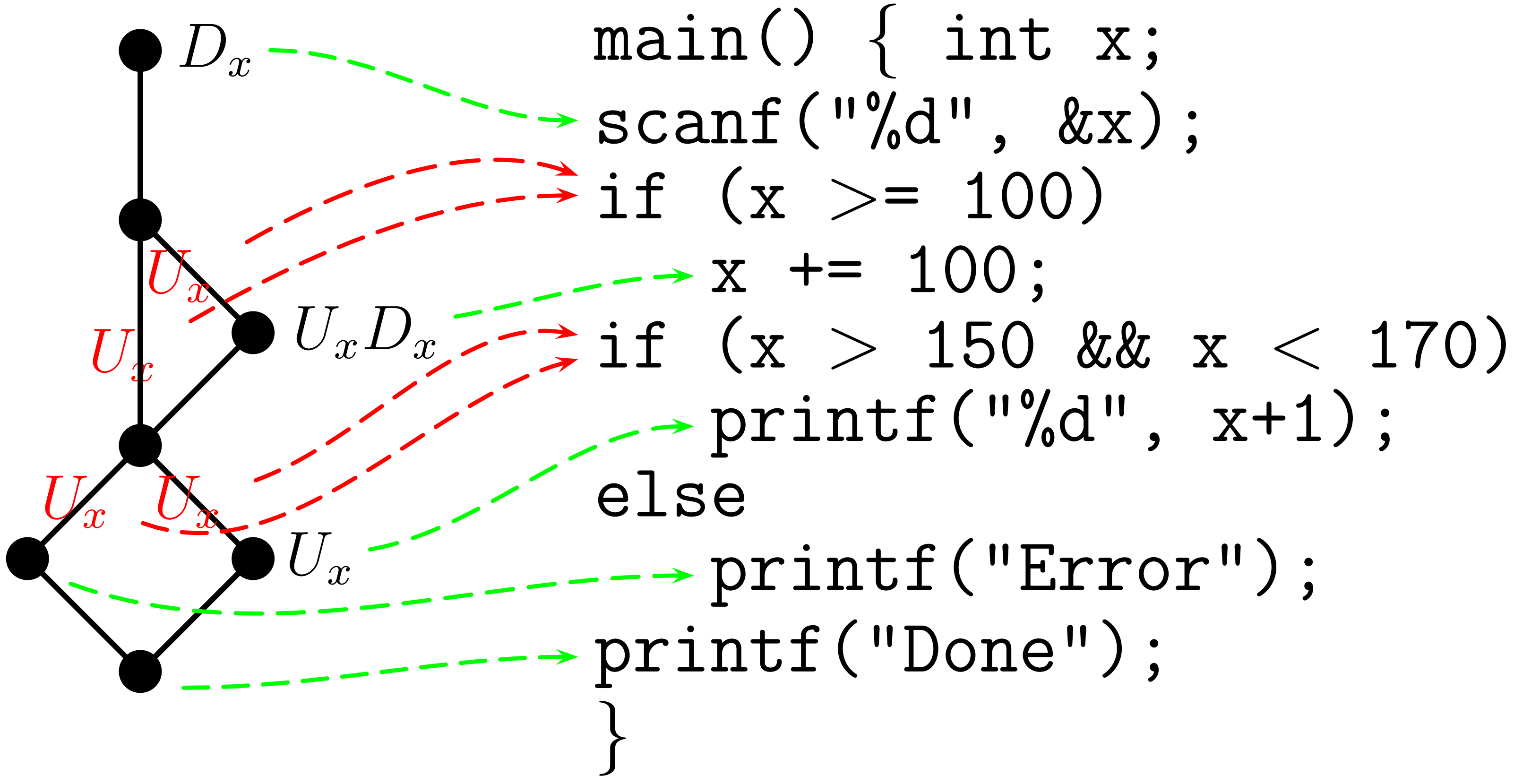


```
main() { int x;
scanf("%d", &x);
if (x >= 100)
    x += 100;
if (x > 150 && x < 170)
    printf("%d", x+1);
else
    printf("Error");
printf("Done");
}
```

Software Modeling

Model

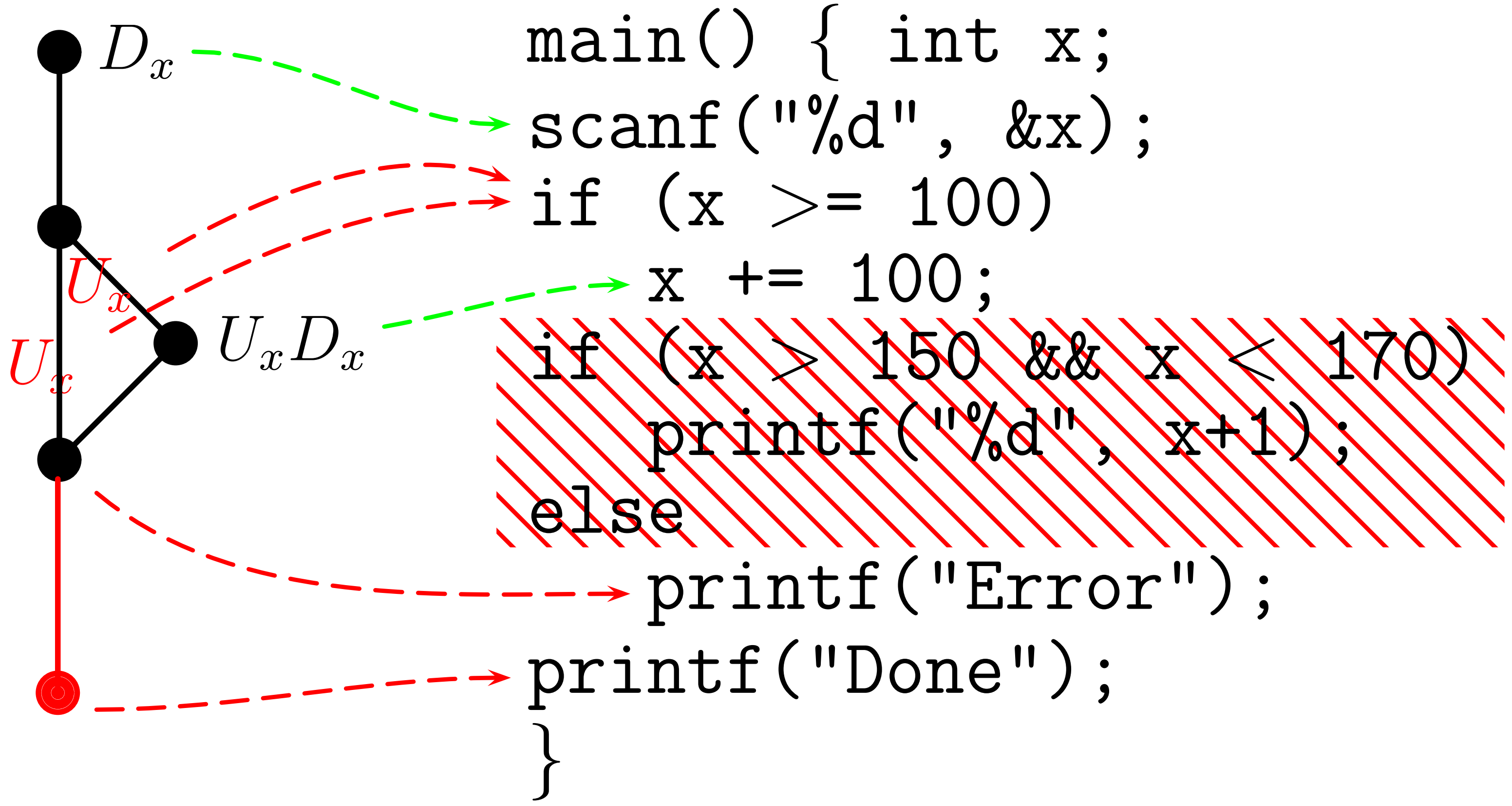
‘Reality’



```
main() { int x;
scanf("%d", &x);
if (x >= 100)
   x += 100;
if (x > 150 && x < 170)
   printf("%d", x+1);
else
   printf("Error");
printf("Done");
}
```

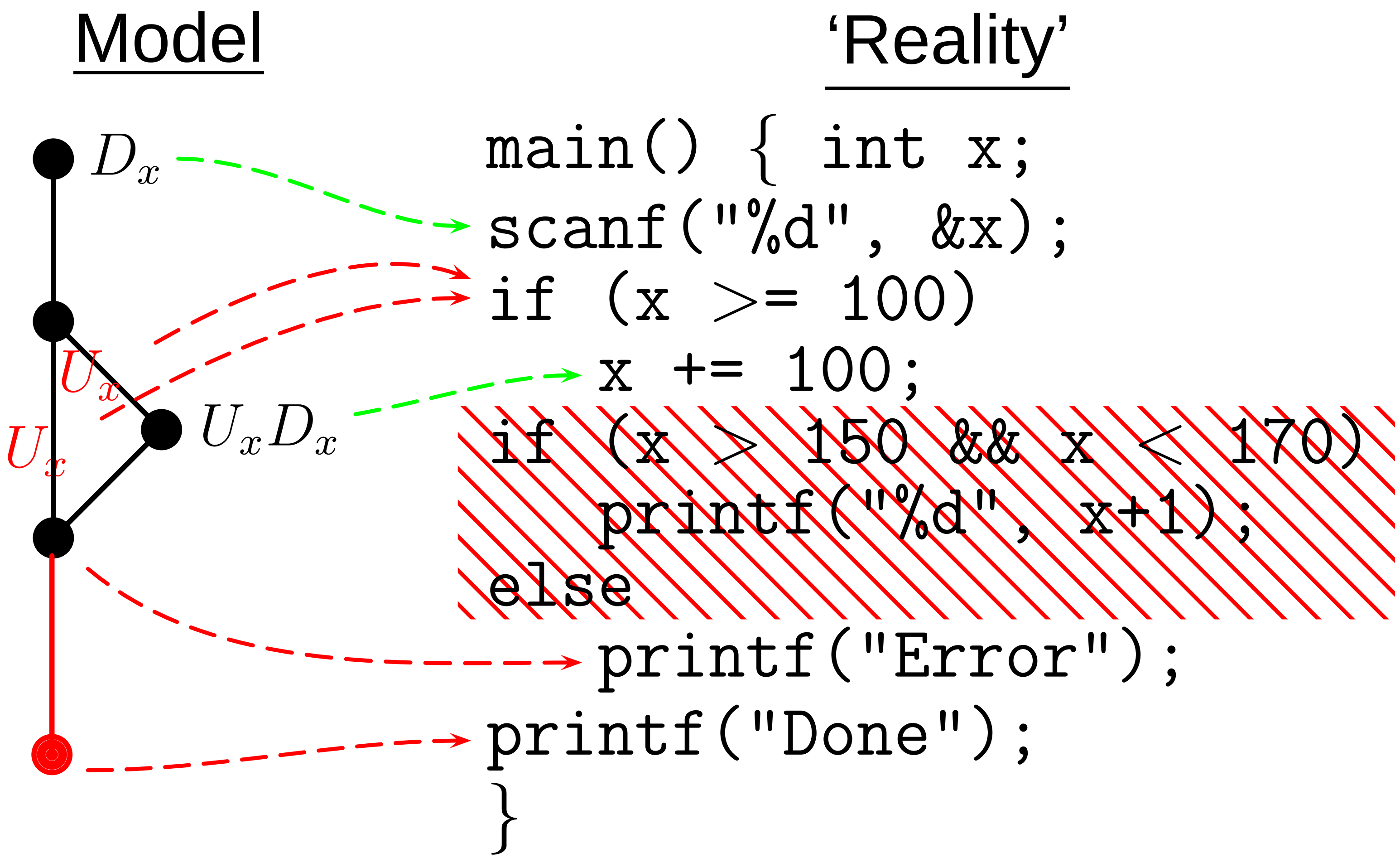
Software Modeling

Model ‘Reality’



```
main() { int x;
scanf("%d", &x);
if (x >= 100)
    x += 100;
if (x > 150 && x < 170)
    printf("%d", x+1);
else
    printf("Error");
printf("Done");
}
```

Software Modeling



```
main() { int x;
scanf("%d", &x);
if (x >= 100)
    x += 100;
if (x > 150 && x < 170)
    printf("%d", x+1);
else
    printf("Error");
printf("Done");
}
```

Initially, the model changes to fit 'reality'
Later, 'reality' is adjusted to fit the model

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What’s To Be Done About It?

Outline of the Talk

I. Philosophy of Software Engineering

II. Mathematics, Science, Engineering

III. And for Software Engineering ... ?

IV. What's To Be Done About It?

The Woes of the Craft

- ▶ Software gets to do the hard parts
  - ▷ The others, limited by natural law, can’t!
- ▶ Crackpot requirements
- ▶ Few professional tools
  - ▷ A builder needs a better saw than a hobbyist – but all Windows are the same
- ▶ Ugly theory (bad mathematics)
- ▶ 60-hour weeks
  - ▷ “We delivered the compiler on time, but none of the marriages survived.”

All Problems Solved by Philosophy?

- Fundamental understanding should help us deal with our difficulties

☞ Software is not subject to natural law ☜

All Problems Solved by Philosophy?

- Fundamental understanding should help us deal with our difficulties

☞ Software is not subject to natural law ☜

- Recommendations:
  - Take responsibility
  - Use good mathematics
  - Keep things straight

Take Responsibility

☞ Without natural law we have no one to blame but ourselves ☜

- ► Civil engineer's "No" backed by physical law
- ► Software engineer's "No" based only on sociology and economics
  - ▷ Crazy requirements and schedules
  - ▷ Releasing untested or failed software
  - ▷ But: keep politics out of "No"
- ► Better tools and working conditions

Use Good Mathematics

☞ Bad mathematics is a matter of choice ☜

- ▶ Safe programming languages 40 years old
  - ▷ Java isn't bad but for the wrong reasons
  - ▷ Giving a software engineer a Turing complete programming language is like giving a child an AK-47

Use Good Mathematics

☞ Bad mathematics is a matter of choice ☜

- ▶ Safe programming languages 40 years old
  - ▷ Java isn't bad but for the wrong reasons
  - ▷ Giving a software engineer a Turing complete programming language is like giving a child an AK-47
- ▶ What about 'Formal Methods'?
  - ▷ Pro: Mathematical theory of a single program – capture it in deep theorems
  - ▷ Con: Mathematics may be all right for expressing God's high-quality laws...

Keep Things Straight

☞ Don’t confuse sociology/economics with mathematics ☜

- Taylorism can hide in software process

  “Management is not a skill or a craft or a profession but a command relationship; a sort of bad habit inherited from the army or the church.”
  –A Lucas Aerospace worker

Keep Things Straight

☞ Don’t confuse sociology/economics with mathematics ☜

- Taylorism can hide in software process

  “Management is not a skill or a craft or a profession but a command relationship; a sort of bad habit inherited from the army or the church.” –A Lucas Aerospace worker

- eXtreme Programming fixes many woes

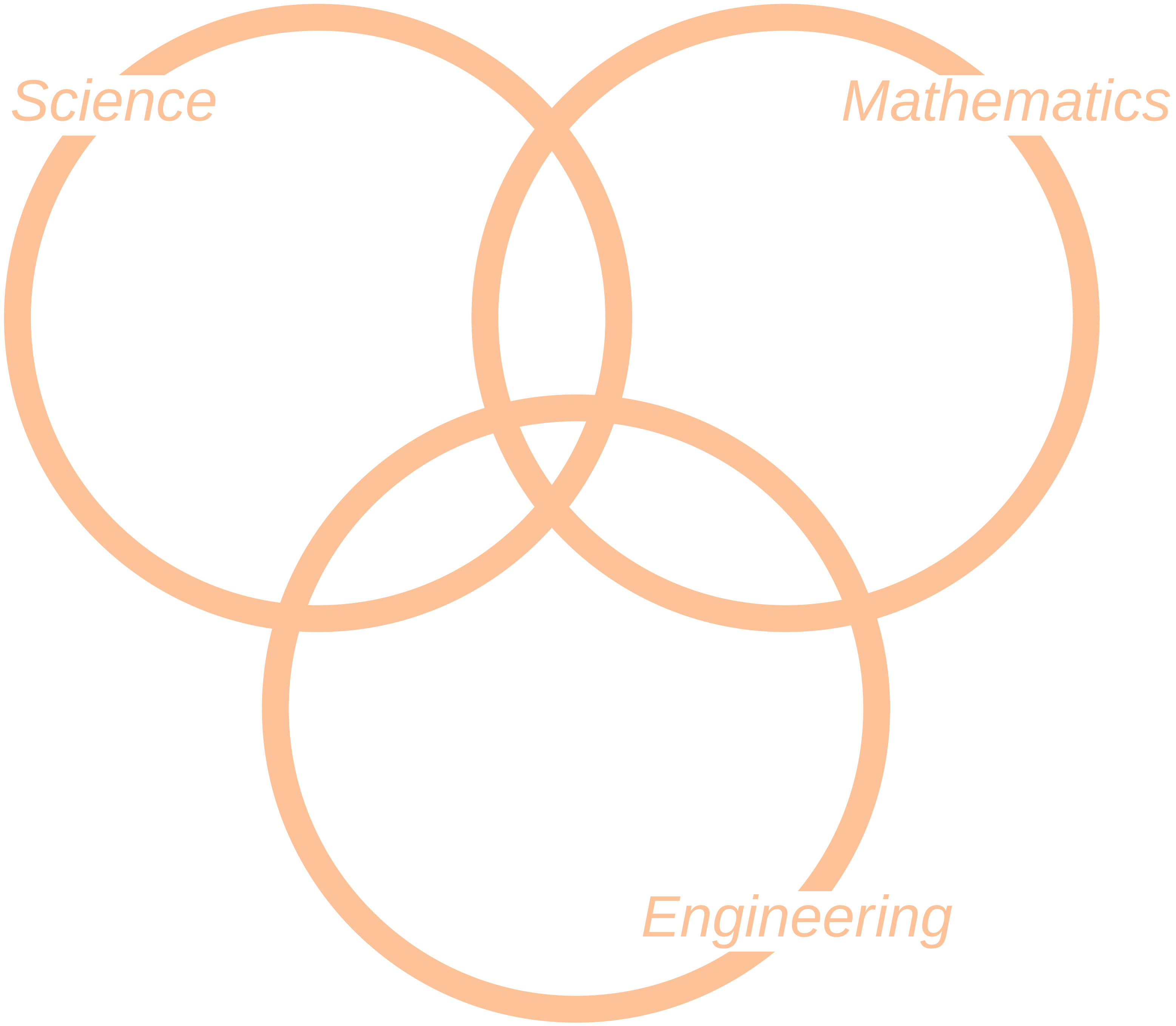
Keep Things Straight

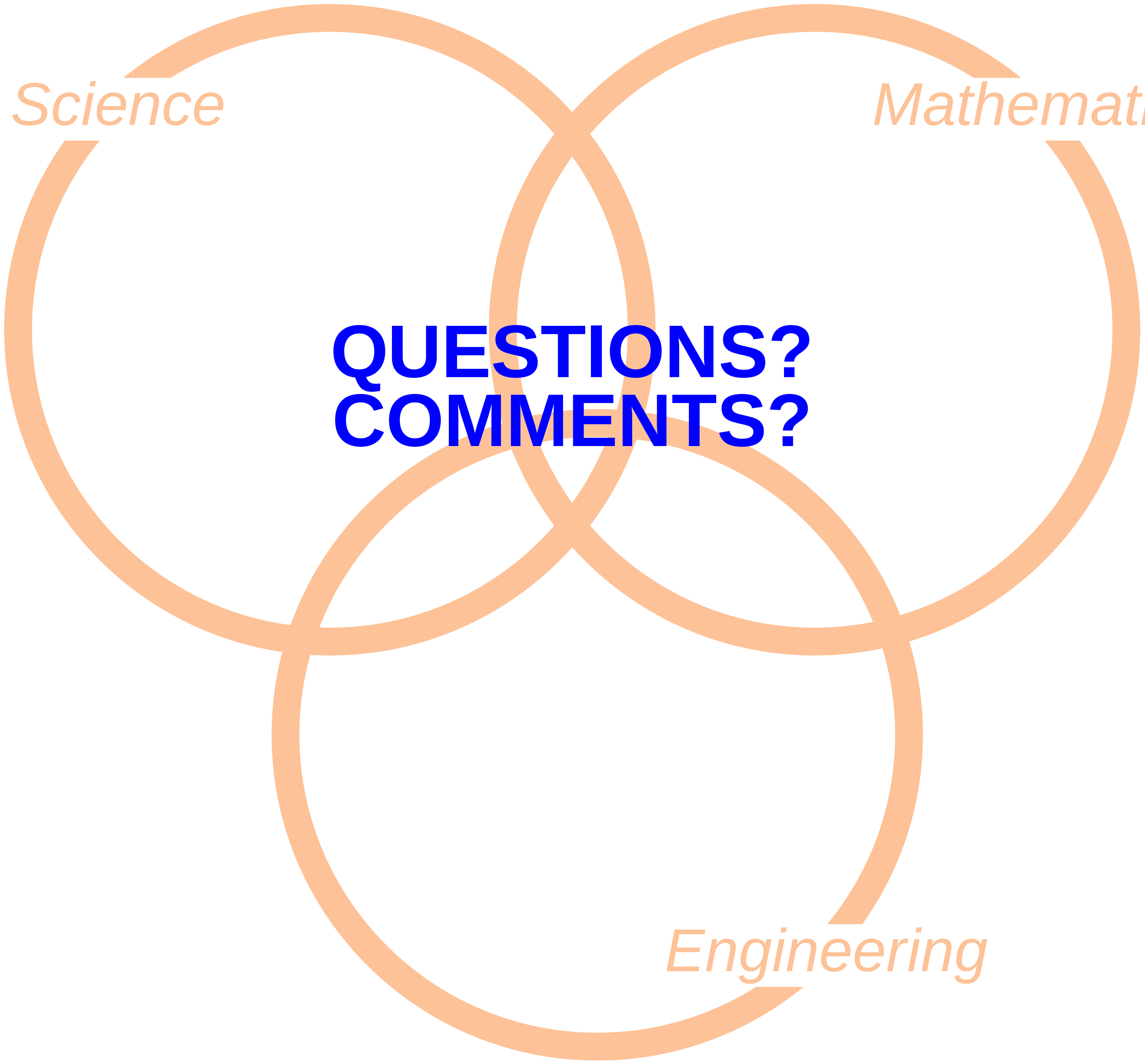
☞ Don’t confuse sociology/economics with mathematics ☜

- Taylorism can hide in software process

  “Management is not a skill or a craft or a profession but a command relationship; a sort of bad habit inherited from the army or the church.”
  –A Lucas Aerospace worker

- eXtreme Programming fixes many woes
- Mathematics doesn’t have to be ‘validated’
  - To model bean counting it isn’t necessary to amass thousands of beans and check $1 + 0 = 1, ..., 2 + 2 = 4, ...$


Science
Mathematics
Engineering



QUESTIONS?
COMMENTS?

Annotated Bibliography

William Addis, *Structural Engineering: The Nature of Theory and Design*, Ellis Horwood, 1991.

- A marvelous book by one of the few philosophers of engineering. Addis speaks as a civil engineer who has studied his discipline historically, and he is not daunted by the immense difficulty of really understanding the past. His purpose is to define engineering design, which he rightly believes is a better name for what engineers do than the overused ‘practice’.

More Bibliography

Thomas S. Kuhn, *The Structure of Scientific Revolutions*, 3rd ed., University of Chicago Press, 1996.

- Kuhn’s thesis is that science has ‘normal’ periods in which a dominant theoretical paradigm enables scientists to work productively, and ‘revolutions’ in which the theoretical paradigm is forced to change. This view of the field is arguably the most influential today. In particular, it has inspired engineers like Addis and Vincenti.

More Bibliography

Karl R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, 5th ed., Routledge, 1992.

- Although Popper and Kuhn do not agree about intrinsic ‘truth’ in science (Kuhn thinks there is none, while Popper still hopes for it), they do agree that the usual descriptions of the so-called scientific method are nonsense. Popper’s view is that theory directs most scientific work, and that science is defined by theories that can be tested and may prove false. Popper believes strongly in the process of rational dispute, and he therefore calls mathematics a science.

More Bibliography

C.A.R. Hoare, “Programming: Science or Sorcery?,” in *Essays in Computing Science*, Prentice-Hall, 1989.

- Hoare presents his vision of a software profession. I would change his ‘science’ to ‘mathematics,’ and his ‘law’ to ‘theorem.’

Walter G. Vincenti, *What Engineers Know and How they Know it*, Johns Hopkins Press, 1990.

- Vincenti does not have Addis’s philosophical turn of mind, but he knows aeronautical engineering and has made a taxonomy of engineering knowledge, with good examples (especially on parameter variation).