# COMP3610/6361
# Principles of Programming Languages

Peter Höfner

Jul 26, 2023

# Section 1

## Introduction

# Foundational Knowledge of Disciplines

## Mechanical Engineering

Students learn about *torque*

$$\frac{\mathrm{d}(r \times \omega)}{\mathrm{d}t} = r \times \frac{\mathrm{d}\omega}{\mathrm{d}t} + \frac{\mathrm{d}r}{\mathrm{d}t} \times \omega$$


Figure: Sydney Harbour Bridge under construction [NMA]

# Foundational Knowledge of Disciplines

## Electrical Engineering / Astro Physics

Students learn about *complex impedance*

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$

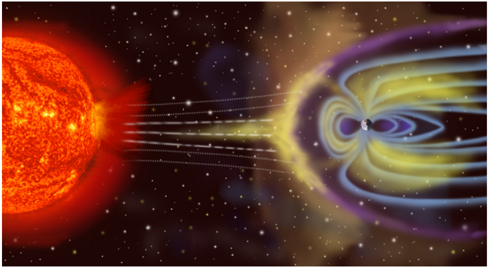

Figure: Geomagnetic Storm alters Earth's Magnetic field [Wikipedia]

# Foundational Knowledge of Disciplines

## Civil Engineering / Surveying

Students learn about *trigonometry*

$$\sin(\theta + \phi) = \sin\theta\cos\phi + \cos\theta\sin\phi$$

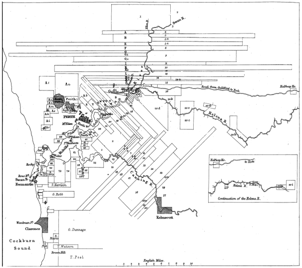


Figure: Surveying Swan River, WA [Wikipedia]

# Foundational Knowledge of Disciplines

**Software Engineering / Computer Science**

Students learn about *???*

Figure: First Ariane 5 Flight, 1996 [ESA]

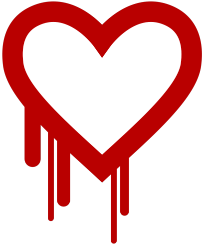

Figure: Heartbleed, 2014 [Wikipedia]

# Programming Languages

**Programming Languages: basic tools of computing**

- what are programming languages?
- do they provide basic laws of software engineering?
- do they allow formal reasoning in the sense of above laws?

# Constituents

- the *syntax* of programs:
  the alphabet of symbols and a description of the well-formed expressions, phrases, programs, etc.
- the *semantics*:
  the meaning of programs, or how they behave
- often also the *pragmatics*:
  description and examples of how the various features of the language are intended to be used

# Use of Semantics

- understand a particular language
  what you can depend on as a programmer;
  what you must provide as a compiler writer
- as a tool for language design:
  - clear language design
  - express design choices, understand language features and interaction
  - for proving properties of a language, eg type safety, decidability of type inference.
- prove properties of particular programs

# Style of Description (Syntax and Semantics)

- natural language
- definition ‘by’ compiler behaviour
- **mathematically**

# Introductory Examples: C

In C, if initially x has value 3, what is the value of the following?

```
x++ + x++ + x++ + x++
```

Is it different to the following?

```
x++ + x++ + ++x + ++x
```

# Introductory Examples: $C^\sharp$

In $C^\sharp$, what is the output of the following?

```
delegate int IntThunk();
class C {
  public static void Main() {
    IntThunk[] funcs = new IntThunk[11];
    for (int i = 0; i <= 10; i++)
    {
        funcs[i] = delegate() { return i; } ;
    }
    foreach (IntThunk f in funcs)
    {
        System.Console.WriteLine(f());
    }
  }
}
```

Australian National University

# Introductory Examples: JavaScript

```
function bar(x) {
    return function () {
        var x = x;
        return x;
    };
}

var f = bar(200);

f()
```

# About This Course

- background: mathematical description of syntax by means of formal grammars, e.g. BNF (see COMP1600)
  clear, concise and precise
- aim I: mathematical definitions of semantics/behaviour
- aim II: understand principles of program design
  (for a toy language)
- aim III: reasoning about programs

# Use of formal, mathematical semantics

**Implementation issues**
Machine-independent specification of behaviour. Correctness of program analyses and optimisations.

**Language design**
Can bring to light ambiguities and unforeseen subtleties in programming language constructs. Mathematical tools used for semantics can suggest useful new programming styles. (E.g. influence of Church's lambda calculus (circa 1934) on functional programming).

**Verification**
Basis of methods for reasoning about program properties and program specifications.

# Styles of semantics

**Operational**
Meanings for program phrases defined in terms of the steps of computation they can take during program execution.

**Denotational**
Meanings for program phrases defined abstractly as elements of some suitable mathematical structure.

**Axiomatic**
Meanings for program phrases defined indirectly via the axioms and rules of some logic of program properties.