



软件科学基础

课程介绍

熊英飞
北京大学

程序出错可能导致灾难性事故



2019年波音737Max坠机事件：埃塞俄比亚航空一架波音737 MAX 8型飞机在起飞阶段坠毁，机上人员全数遇难。

2016年特斯拉车祸：自动驾驶模式下的特斯拉汽车和卡车相撞，导致驾驶员当场丧生

2011年亚马逊宕机事故：亚马逊云计算出现了超过2天的宕机事故，造成的资金和信誉损失难以估算



事故原因：飞机MCAS防失速自动系统软件存在缺陷

事故原因：在强烈日光条件下，摄像头进入盲区，但软件系统并没有捕获这一情况

事故原因：软件配置错误导致部分结点请求激增，不断转发请求压垮网络



如何知道程序是正确的？



程序员

给我写一个排序

```
写好了，看：  
quicksort (x:xs) =  
  quicksort [a | a <- xs, a <= x] ++ [x]  
  ++ quicksort [a | a <- xs, a > x]
```




老板

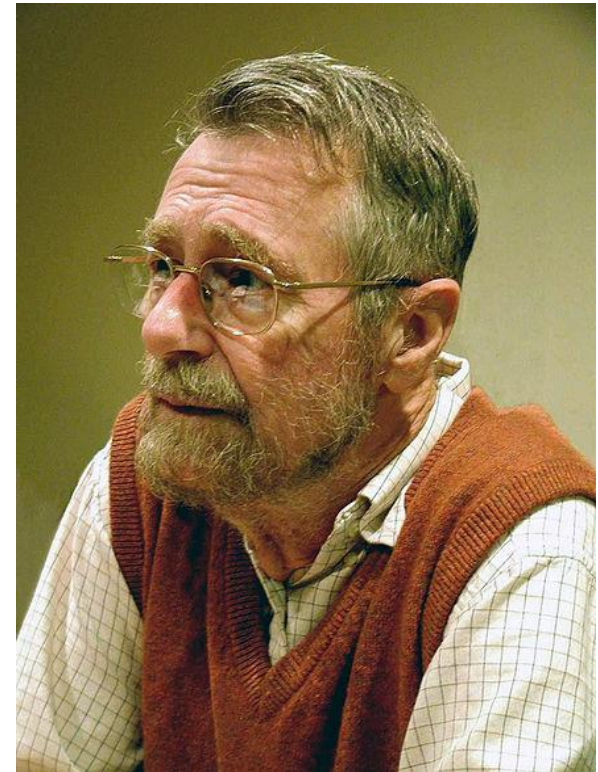
写对了吗？



如何知道程序是正确的？

-  程序员：我测试了！

Testing shows the presence,
not the absence of bugs.



Edsger W. Dijkstra



如何知道程序是正确的？



- 程序员：你看，我先把比 x 小的选出来排序，然后把比 x 大的选出来排序，然后把三部分按顺序合起来，过程多么合理！
 - 自然语言存在二义性
 - 资深程序员：“排序”的要求是什么？升序还是降序？要求时间复杂度吗？排序元素的定义域和序列长度通常哪个大？
 - 数学家：“排序”的定义是什么？被排序元素上存在全序吗？存在偏序吗？
 - 不容易判断自然语言的分析是否全面、正确
 - 上述描述没有分析终止性和边界条件

“牌序”？好像是挺重要的





如何解决自然语言论证的问题？

- 数学：对事物的抽象结构与模式进行严格描述的一种通用手段（百度百科）
 - 解决二义性问题
- 逻辑学：研究推理的学科，即研究如何从前提必然推出结论（百度百科）
 - 解决论证正确性问题
- 数理逻辑：严格描述的逻辑学，是现代数学的基础
 - 一系列语法规则，用于描述命题
 - 一系列推理规则，用于证明命题是否成立



课程内容1：数理逻辑

- 高阶逻辑：
 - 与、或、非、全称量词、存在量词、谓词
- 基本的推理规则
 - 演绎
 - 归纳

假如我早年在数理逻辑上好好下点功夫的话，我就不会犯这么多的错误。不少东西逻辑学家早就说了，可我不知道。要是我能够年轻20岁的话，就要回去学逻辑。



Edsger W. Dijkstra

如何用数理逻辑来论证程序正确性?



- 程序员:
 - 用数理逻辑证明如下定理:
 - $\forall l, i: bag(l) = bag(quick\text{sort}(l)) \wedge quick\text{sort}(l)[i - 1] \leq quick\text{sort}(l)[i]$
- 稍等, 什么是 $quick\text{sort}$ 函数?
 - 之前只写了一段 $quick\text{sort}$ 程序

如何用数理逻辑来论证程序正确性?



- 如何精确的定义程序?
 - 静态：如何定义什么是程序?
 - 形式语法
 - 动态：如何定义程序在执行时的行为?
 - 形式语义
- 能否在程序上直接推理?
 - 霍尔逻辑：关于论证程序行为的逻辑



课程内容2：形式语义

- 形式语法：上下文无关文法
- 形式语义：
 - 操作语义：将语句解释为抽象机器上的操作
 - 公理语义：将语句解释为逻辑系统中的推导规则
- 公理语义又叫霍尔逻辑



论证程序的正确性

- 老板：我们公司的1000万行程序都需要证明正确性
- 困难1：人力时间成本
 - seL4：证明功能正确的操作系统内核
 - 写代码用了2.2人年，写证明用了20人年
- 困难2：怎么知道证明写对了
 - <https://www.win.tue.nl/~gwoegi/P-versus-NP.htm>
 - 至少有62篇论文证明了 $P = NP$ ，50篇论文证明了 $P \neq NP$



针对困难1：能不能让计算机自动判断程序的正确性？

- 能否让计算机自动证明程序正确性或不正确性？

否定三联



哥德尔

“总是有些定理不存在证明的。”
——哥德尔不完备定理，1931年

“对于停机这个性质，无论什么算法，总是有程序没法自动证的。” ——停机问题，1936年



莱斯

“世界上绝大多数程序性质都跟停机一样没法自动证。” ——莱斯定理，1953年



图灵

妥协：自动证不出来的程序 就不让写



- 类型系统：类型系统采用自动分析阻止程序犯某种错误
 - C语言的类型系统阻止了什么错误？（相对B语言）
 - Java的类型系统包含throws关键字，阻止了什么错误？
- 类型系统通常不能准确判断任意程序，会禁止部分正确程序的编写
 - 能否举一个被C语言类型系统阻止的正确程序的例子？
 - `int a = 1; int b = &a; int* c = b; return *c;`
 - 能否举一个被Java异常检查系统阻止的正确程序的例子？
 - `void m() throws IOException {
 if (false) throw new DataFormatException(); }`
- 考虑语言的表达能力，目前类型系统只能处理很小一部分错误类别
 - 部分高效算法已经无法在Rust写出



课程内容3： 类型系统

- 类型系统基本概念
- 小型带类型语言STLC
- 引用类型
- 子类

针对困难2：能否自动检查证明的正确性？



- 能，并且能套用类型检查算法

Haskell Brooks Curry



“‘命题-证明’和‘类型-值’之间存在对应关系。”
——Curry-Howard Correspondence, 1934-1969



课程内容4： 交互式定理证明工具

- 交互式定理证明工具Coq
 - 包含函数式编程语言Gallina
 - 支持定义命题和证明
 - 自动检查证明是否证明命题
 - 并随时提示程序员还没有完成证明的部分
 - 前三部分课程内容均可在Coq中写出

小结：为程序正确性构建的理论和方法



- 数理逻辑：定义和证明定理
- 形式语义：证明程序的正确性
- 类型系统：阻止部分类型的错误
- 交互式定理证明工具：确保证明的正确性



计算机理论分类

- 理论计算机科学通常根据ICALP的CFP分成A， B两类

Track A: Algorithms, Complexity and Games

- * Algorithmic and Complexity Aspects of Network Economics
- * Algorithmic Aspects of Networks and Networking
- * Algorithmic Aspects of Security and Privacy
- * Algorithms for Computational Biology
- * Algorithmic Game Theory and Mechanism Design
- * Approximation and Online Algorithms
- * Combinatorial Optimization
- * Combinatorics in Computer Science
- * Computational Complexity
- * Computational Geometry
- * Computational Learning Theory
- * Cryptography
- * Data Structures
- * Design and Analysis of Algorithms
- * Distributed and Mobile Computing
- * Foundations of Machine Learning
- * Graph Mining and Network Analysis
- * Parallel and External Memory Computing
- * Quantum Computing
- * Randomness in Computation
- * Theoretical Foundations of Algorithmic Fairness

Track B: Automata, Logic, Semantics, and Theory of Programming

- * Algebraic and Categorical Models of Computation
- * Automata, Logic, and Games
- * Database Theory, Constraint Satisfaction Problems, and Finite Model Theory
- * Formal and Logical Aspects of Learning
- * Formal and Logical Aspects of Security and Privacy
- * Logic in Computer Science and Theorem Proving
- * Models of Computation: Complexity and Computability
- * Models of Concurrent, Distributed, and Mobile Systems
- * Models of Reactive, Hybrid, and Stochastic Systems
- * Principles and Semantics of Programming Languages
- * Program Analysis, Verification, and Synthesis
- * Type Systems and Typed Calculi

- A为计算的理论， 也被部分国内学者称为美式理论计算机科学
- B为软件的理论， 也被部分国内学者称为欧式理论计算机科学
- 保障软件正确性的理论构成了现代软件系统的基础， 也组成了理论计算机科学的半边天



本课程与相关课程

- 本课程：数理逻辑、形式语义、类型系统、Coq
- 计算概论A实验班（本，胡振江、张伟）
 - 系统学习函数式程序设计和证明语言Agda
- 数理逻辑（本，王捍贫）、高级逻辑学（研，谢冰）
 - 深入学习数理逻辑
- 程序设计语言的形式语义（研，王捍贫、曹永知）
 - 深入学习形式语义
- 编程语言设计原理（研，胡振江、赵海燕）
 - 深入学习类型系统



本课程与相关课程

- 本课程：数理逻辑、形式语义、类型系统、Coq
- 软件分析技术（本，熊英飞）
 - 自动判断部分程序正确性的理论、方法和技术
- 软件测试导论（本，郝丹）
 - 自动判断部分程序不正确性的理论、方法和技术
- 编译原理（本，张路、刘先华、王迪等）
 - 如何实现程序设计语言
- 概率编程导论（研，张昕）
 - 如何基于概率构建程序设计语言



教学方式

- 传统理论课的问题：在理论课上得高分的同学仍然存在理论基础不牢的问题
 - 概念不清、混用数学概念和编程概念、证明推导随意
- 猜想：数学定理证明不好阅卷，采用“显然”“易证”等模糊说法也能过关
- 本课程：首先介绍Coq，所有理论在Coq中构建，所有证明习题用Coq完成
 - 理论基础：数理逻辑、形式语义、类型系统
 - 实践：在Coq中实现上述内容和证明定理

教材



网址：<https://softwarefoundations.cis.upenn.edu/>

志愿者维护的中文翻译版（不推荐）：<https://coq-zh.github.io/SF-zh/>

课程采用最新版英文教材



作业形式

- 教材每一章都是一个Coq文件，教材正文为注释，大部分习题为不完整的Coq程序
- 将Coq程序补充完整，运行脚本自我打分
- 本地通过之后发送文件给助教



预期学习收益

- 打基础
 - 对软件理论知识有全面了解
 - 为进一步学习后续课程打下基础
 - 为开展软件方向的科研打下基础
- 增能力
 - 熟练使用交互式定理证明编程语言和工具
 - 对软件理论的了解提升日常软件开发能力
- 找工作
 - 带证明软件于2011年被《麻省理工技术评论》评为年度十大技术
 - 越来越多的软件企业开始采用定理证明的方式构造核心代码
 - 华为、阿里、微软
 - 部分航空航天、高铁企业、研究所
 - 自动定理证明也是大模型企业专攻的方向
 - DeepMind, OpenAI, DeepSeek.....

总裁办电子邮件

电邮通知【2019】068号 签发人：任正非

关于对部分2019届顶尖学生实行年薪制管理的通知

华为公司要打赢未来的技术与商业战争，技术创新与商业创新双轮驱动是核心动力，创新就必须要有世界顶尖的人才，有顶尖人才充分发挥才智的组织土壤。我们首先要用顶级的挑战和顶级的薪酬去吸引顶尖人才，今年我们先将从全世界招进20-30名天才“少年”，今后逐年增加，以调整我们队伍的作战能力结构。

经公司研究决定，对八位2019届顶尖学生实行年薪制，年薪制方案如下：

1. 钟利，博士。
年薪制方案：182-201万人民币/年
2. 梁通，博士。
年薪制方案：182-201万人民币/年
3. 李屹，博士。
年薪制方案：140.5-156.5万人民币/年
4. 管高扬，博士。
年薪制方案：140.5-156.5万人民币/年
5. 曹许业，博士。



课程主页

- <https://xiongyingfei.github.io/SF/>



软件科学基础 / 2024春

课程简介

理论计算机科学通常被认为分为A和B两部分（参考ICALP的CFP），A主要是计算相关的理论，包括自动机、算法复杂度等；B主要是软件相关的理论，包括数理逻辑、形式语义、类型系统等。部分国内学者也通俗的将这两部分称为“美式计算机科学”和“欧式计算机科学”。本课程主要讲授软件部分的基础理论，即数理逻辑、形式语义、类型系统的知识。

这些理论知识不仅能丰富和加深我们对软件的理解，也是构造高可信软件的基础。为了确保程序是可信的，一种本质手段是将证明嵌入程序，让计算机自动检查该证明的正确性，确保程序满足需求规约。这样的“带证明程序代码”是目前很多大公司构造高可信关键代码的基础，比如微软、华为都用这种技术构造操作系统内核，而各国的航天航空、高铁等关键代码也常常采用带证明代码的方式构造。该技术也入选了《麻省理工技术评论》2011年选出的年度十大技术。本课程的实践部分将介绍构造这类代码的一个基础编程语言Coq，使得同学们能够掌握构造高可信软件的基本方法，并通过动手实践加深对软件理论的理解。

本课程将首先讲授Coq定理证明工具的使用，然后基于Coq介绍数理逻辑、形式语义、类型系统等相关知识，并在Coq中构建一个程序设计语言并证明该语言实现的各种性质。

课程信息

上课时间：双周二7-8节、每周周四5-6节
上课地点：第三教学楼 401

参考书目

《Software Foundations》，在线书籍

- 网址：<https://softwarefoundations.cis.upenn.edu/>
- 由志愿者维护的中文翻译版：<https://coq-zh.github.io/SF-zh/>

本课程将用到该书籍的第一卷*Logic Foundations*和第二卷*Programming Language Foundations*。

在线书籍实时更新，本课程本年讲授的时候将采用最新版书籍。





任课教师与助教

- 教师：熊英飞
 - 2009年于日本东京大学获得博士学位
 - 2009-2011年在加拿大滑铁卢大学从事博士后研究
 - 2012年加入北京大学，现任长聘副教授
 - 办公室：理科一号楼1431
 - 邮件：xiongyf@pku.edu.cn
- 助教：黄柘铨
 - 博士一年级
 - 邮件：willhuang@stu.pku.edu.cn



评分方式

- 作业：50分
- 期末考试：50分
- 作业：
 - 独立完成
 - 下周四上课前提交
 - 助教评分
- 考试：
 - 闭卷考试
 - 考察概念理解和运用为主
 - 尽量避免需要记忆的内容，如果考到会给出
 - 难度控制：尽量做到平时搞懂学习内容，独立完成习题的情况下可以得高分



开课历史和评估分数

年份	课程评分
2021 (研究生, 软件理论基础与实践)	98.89
2022 (本科生, 软件理论基础与实践)	96.67
2023 (本科生, 软件科学基础)	97.4
2024 (本科生, 软件科学基础)	95.83

10. 您对课程、教师和教学的意见与建议(评估人数:2)

序号	内容
1	无
2	课程内容非常丰富, 可以为今后做理论研究的人打下很好的基础; 老师备课认真, 讲课风格幽默, 互动性强。绝对是精品课程(▽▽)
1	课讲的好! 难度适中, 覆盖知识点广, 课程内容有着切合实际的背景, 讲解非常自然。给授课老师和助教点赞!
2	好课! 希望增加学分、学时、课时、课程内容!
1	希望能扩展到4学分8学时, 把分离逻辑也讲完。
2	感觉课上多讲一些理论东西比较好而不是代码演示, 在大屏幕上看到代码很难受, 而且教材对代码也介绍得足够详细了。可以侧重constructive logic和Type theory的讲解, 代码部分让同学课后完成。
3	老师有时候带着学生一步一步证明是很易懂的, 希望老师可以更多以这样的方式教学



作业

- 下载教科书及相关Coq代码
 - <https://softwarefoundations.cis.upenn.edu>
- 安装Coq系统和至少一个开发环境
 - <https://coq.inria.fr/download>
 - CoqIDE：自带独立开发环境
 - VSCode插件：VSCoq
 - Emacs插件：Proof General