Discrete Mathematics
(for Computer Science)

Zhaoguo Wang, Zeyu Mi

IPADS实验室

Why Choose CDM?

4 yrs in SJTU

For graduation

Graduate school

Interview

Startup

Programmer
/ Researcher

Why Choose CDM?

V.S.

What makes you distinguished?

Why Choose CDM?

V.S.

What makes you distinguished?

Understand “laws of physics” in CS

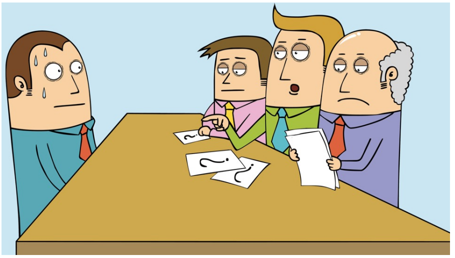
Be able to answer the fundamental questions.

Questions To Be Answered In CDM

What problems can you solve with a computer?

How can you be certain of your answer to these questions?

How Do These Two Questions Affect Each Path?



Interview

Interview Question:
what languages r u good at?

Java, Java Spring, Ruby, Python

Java, C++, .Net, C, Go, Rust

Besides above all, assembly language, Coq, TLA and so on…

How Do These Two Questions Affect Each Path?

Graduate school

Major concern by Profs.

Is the student a logical person?

He got a grade of A for CDM.

He can have logical thinking and answer the questions in a very logical way.

How Do These Two Questions Affect Each Path?

Startup

When talk to investors

The first feeling is important

His story makes sense.

He has a very clear mind.

How Do These Two Questions Affect Each Path?

Programmer / Researcher

How do you know your program is correct?

There is no segmentation fault.

It passes all tests.

I can prove it.

How To Answer These Two Questions?

What problem can you solve with a computer?

How can you be certain of your answer to these questions?

大纲

- 图论
  - 图的基本概念
  - 道路与回路
  - 树
- 逻辑
  - 命题逻辑
  - 谓词逻辑
  - 公理系统
- 集合论
  - 集合
  - 关系
  - 函数

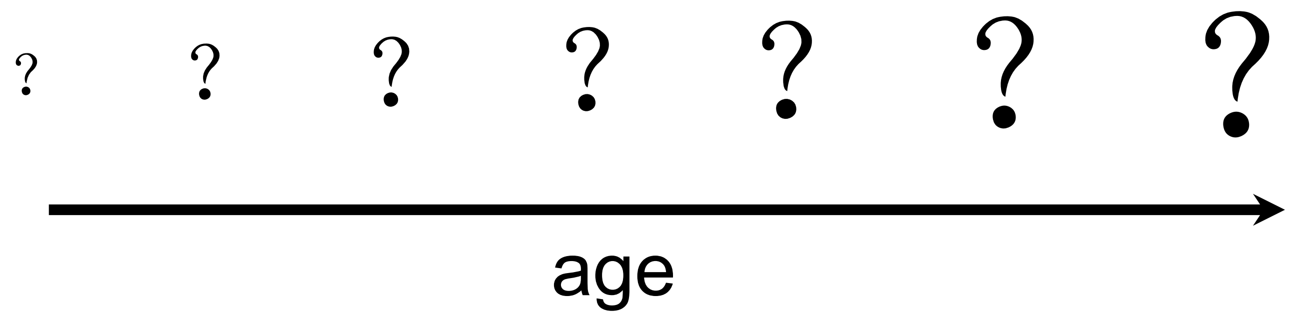
大纲

- 图论
  - 图的基本概念
  - 道路与回路
  - 树
- 逻辑
  - 命题逻辑
  - 谓词逻辑
  - 公理系统
- 集合论
  - 集合
  - 关系
  - 函数

?

大纲

- 图论
  - 图的基本概念
  - 道路与回路
  - 树
- 逻辑
  - 命题逻辑
  - 谓词逻辑
  - 公理系统
- 集合论
  - 集合
  - 关系
  - 函数



大纲

- 图论
  - 图的基本概念
  - 道路与回路
  - 树
- 逻辑
  - 命题逻辑
  - 谓词逻辑
  - 公理系统
- 集合论
  - 集合
  - 关系
  - 函数

这将不是一门普通的离散数学

? ? ? ? ? ? ?

age

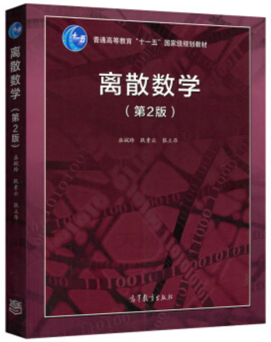
Be Practical than Attractive

Is it solvable?

```
bool is_power2(unsigned int v)
{
    unsigned int i;
    for(i = 0; i <= 31; ++i)
    {
      if(v == (1 << i))
         return true;
    }
    return false;
}
```

How to prove its correctness?

1983



2015

离散数学 =

- 图论
- 数理逻辑
- 集合论
- 组合数学
- ……

What we teach:

Discrete Mathematics
for **Computer Science**

NYU

RICE
Unconventional Wisdom

- Automata
- Turing machine
- Computation theory

- Satisfiability problem
- Satisfiability modulo theory

How to module a program

How to prove a program

Three Parts of CDM

Part I. Reasoning. (~*10 Weeks*)

How to prove the correctness?

Part II. Computability. (~ *5 Weeks*)

Is the problem computable (solvable)?

Part III. Applications. (~ *1 Weeks*)

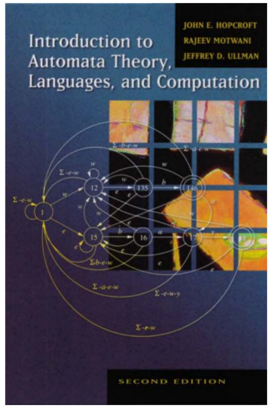
Solve the problem in real systems.

References

Textbooks

《数理逻辑与集合论》第2版

- 石纯一 著，清华大学出版社



John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation, Pearson, 2001

To Be Distinguished, You Need To

Take

- ✓ Lectures
- ✓ Recitation (Optional)

Do

- ✓ 3 Labs
- ✓ Homework (Preview + Review)

Pass

- ✓ Quiz
- ✓ Final exam

Grading Breakdown

- Lecture + Homework + Quiz + (Optional Lab) 35%
- Lab 25%
- **Final 40%**

Policies

You must work alone on all assignments

- You may post questions on Canvas.
- You are encouraged to answer others’ questions, but refrain from explicitly giving away solutions.

Labs & Exercises

- Assignments due at 11:59pm on the due date
- Everybody has 5 grace days
  - Must make the claim before the due
- Zero score after the due

Integrity and Collaboration Policy

We will enforce the policy strictly.

1. The work that you turn in must be yours
2. You must acknowledge your influences
3. You must not look at, or use, solutions from prior years or the Web, or seek assistance from the Internet
4. You must take reasonable steps to protect your work
   - You must not publish your solutions
5. If there are inexplicable discrepancies between exam and lab performance, we will over-weight the exam and interview you.

Faculty Information

Lecturer
Zhaoguo Wang (王肇国)
zhaoguowang@sjtu.edu.cn

Lecturer
Zeyu Mi (糜泽羽)
yzmizeyu@sjtu.edu.cn

TA
Haoning Lan (蓝浩宁)
sjtulhn@sjtu.edu.cn

TA
Zhuohao Shen (申倬豪)
ao7777@sjtu.edu.cn

Class Info

Website:
http://ipads.se.sjtu.edu.cn/courses/cdm

Canvas:
https://oc.sjtu.edu.cn/courses/47984

2022离散数学课程群

该二维码7天内(9月16日前)有效，重新进入将更新

Three Parts of CDM

Part I. Reasoning.

Part II. Computability.

Part III. Applications.

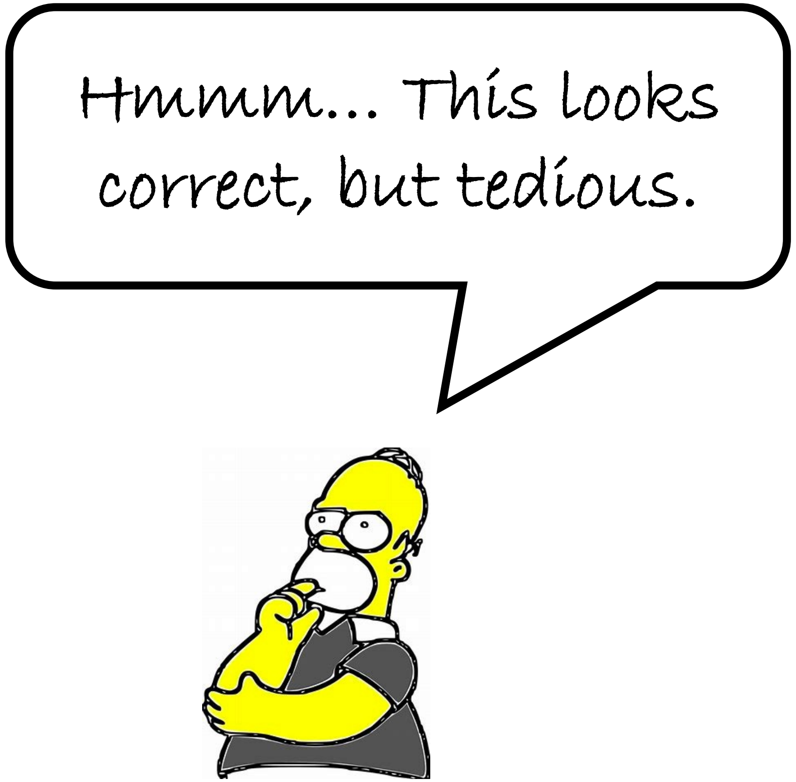
Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    unsigned int i;
    for(i = 0; i <= 31; ++i)
    {
      if(v == (1 << i))
          return true;
    }
    return false;
}
```

Q: What is the semantic of this piece of code?

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    unsigned int i;
    for(i = 0; i <= 31; ++i)
    {
      if(v == (1 << i))
          return true;
    }
    return false;
}
```



Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = !(v & (v - 1));
    return f;
}
```

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = !(v & (v - 1));
    return f;
}
```

Q: How to ensure the correctness?

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = !(v & (v - 1));
    return f;
}
```

Q: How to ensure the correctness?

Run some test cases?

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = !(v & (v - 1));
    return f;
}
```

What if v is 0?

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = v && !(v & (v - 1));
    return f;
}
```

Fix the bug.

Hmmm… Testing can find some bugs, but can not prove its correctness.

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = v && !(v & (v - 1));
    return f;
}
```

Can we prove it?

Hmmm… Let me have a try with a pen. Maybe I can prove it with natural language.

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = v && !(v & (v - 1));
    return f;
}
```

Can we prove it?

A Manual Proof

V is a 32-bits integer. V is a power of 2 iff there is one and only one bit is 1.
In “v && !(v & (v-1))”, v ensures there is one 1-bit and !(v & (v-1)) ensures there is only one 1-bit.

Part I: Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = v && !(v & (v - 1));
    return f;
}
```

Can you ensure your logic and every sentence is correct?

Hmmm... Kill me...

Part I: Computer Aided Reasoning

```
bool is_power2(unsigned int v)
{
    bool f;
    f = v && !(v & (v - 1));
    return f;
}
```

Ask the computer to prove
its correctness...
What should we do first?

Part I: Computer Aided Reasoning

```
bool xor(unsigned int v)
{
    return v ^ v;
}
```

$$(((\neg v1) \land v1) \lor (v1 \land (\neg v1)))\land$$
$$(((\neg v2) \land v2) \lor (v2 \land (\neg v2)))\land$$
$$(((\neg v3) \land v3) \lor (v3 \land (\neg v3)))\land$$
$$\ldots$$
$$(((\neg v32) \land v32) \lor (v32 \land (\neg v32)))$$

A Propositional Formula

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.

Part I: Computer Aided Reasoning

```
void swap(int a, int b)
{
       a ^= b;
       b ^= a;
       a ^= b;
}
```

$$
\begin{array}{l}
(a1 \leftrightarrow (((((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1))) \wedge (\neg b1)) \vee \\
((\neg(((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1)))) \wedge b1))) \wedge \\
(b1 \leftrightarrow (((((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1))) \wedge \\
(\neg((((((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1))) \wedge (\neg b1)) \vee \\
((\neg(((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1)))) \wedge b1))))) \vee \\
((\neg(((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1)))) \wedge \\
((((((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1))) \wedge (\neg b1)) \vee \\
((\neg(((a1 \wedge (\neg b1)) \vee ((\neg a1) \wedge b1)))) \wedge b1)))))) \wedge \\
\ldots
\end{array}
$$

A Propositional Formula

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.

It is complicated for both human being and computer.

Part I: Computer Aided Reasoning

```
void swap(int a, int b)
{
        a ^= b;
        b ^= a;
        a ^= b;
}
```

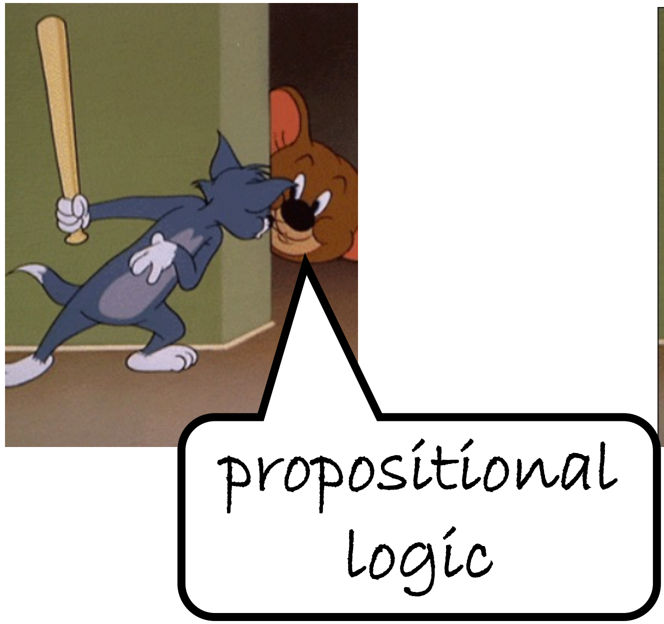
$$a = XOR(b, XOR(a, b)) \wedge$$
$$b = XOR(XOR(a, b), XOR(b, XOR(a, b)))$$

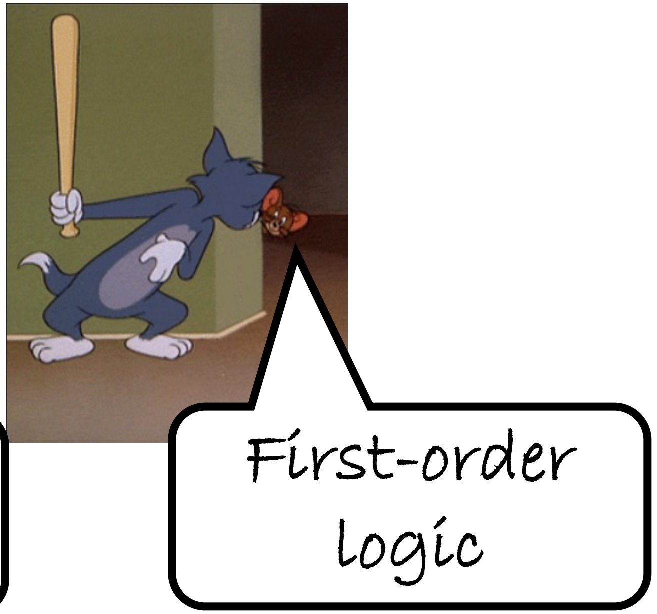
A Predicate Formula

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.





Part I: Computer Aided Reasoning

```
void swap(int a, int b)
{
    a ^= b;
    b ^= a;
    a ^= b;
}
```

$$a = XOR(b, XOR(a, b)) \wedge$$
$$b = XOR(XOR(a, b), XOR(b, XOR(a, b)))$$

A Predicate Formula

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first logic formula.

Step 2. Ask the computer to solve the formula.

Part I: Computer Aided Reasoning

```
int clone(unsigned int n)
{
    int i = 0;
    while(i < n)
    {
        i = i + 1;
    }
    return i;
}
```

Loop can not be addressed by first-order logic.

Can we tell computer this is a loop?

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first logic formula.

Step 2. Ask the computer to solve the formula.

Part I: Computer Aided Reasoning

```
int clone(unsigned int n)
{
    int i = 0;
    while(i < n)
    {
        i = i + 1;
    }
    return i;
}
```

➡

```
method clone(n: nat) returns (b : nat)
  ensures b == n
{
   var i := 0;
   while i < n
      invariant 0 <= i
   {
      i := i + 1;
   }
   return i;
}
```

1.1 Propositional Logic

Step 1. Convert it into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants

Part I: Computer Aided Reasoning

```
int clone(unsigned int n)
{
    int i = 0;
    while(i < n)
    {
        i = i + 1;
    }
    return i;
}
```

➡

```
method clone(n: nat) returns (b : nat)
  ensures b == n
{
   var i := 0;
   while i < n
      invariant 0 <= i
   {
      i := i + 1;
   }
   return i;
}
```

		Description	Line	Column
⊗	1	A postcondition might not hold on this return path.	10	3
	2	This is the postcondition that might not hold.	2	12

Part I: Computer Aided Reasoning

```
int clone(unsigned int n)
{
    int i = 0;
    while(i < n)
    {
        i = i + 1;
    }
    return i;
}
```

➡

```
method clone(n: nat) returns (b : nat)
  ensures b == n
{
   var i := 0;
   while i < n
      invariant 0 <= i && i <= n
   {
      i := i + 1;
   }
   return i;
}
```

```
Dafny 2.3.0.10506

Dafny program verifier finished with 1 verified, 0 errors
Program compiled successfully
```

Part I: Computer Aided Reasoning Overview

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

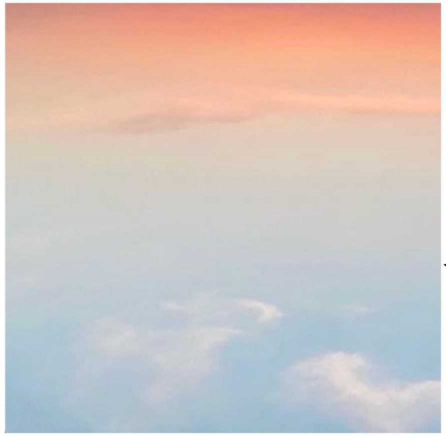
Step 1. Convert program into first logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants



3 Labs in total.
Good luck to all!

TA
Haoning Lan(蓝浩宁)
sjtulhn@sjtu.edu.cn