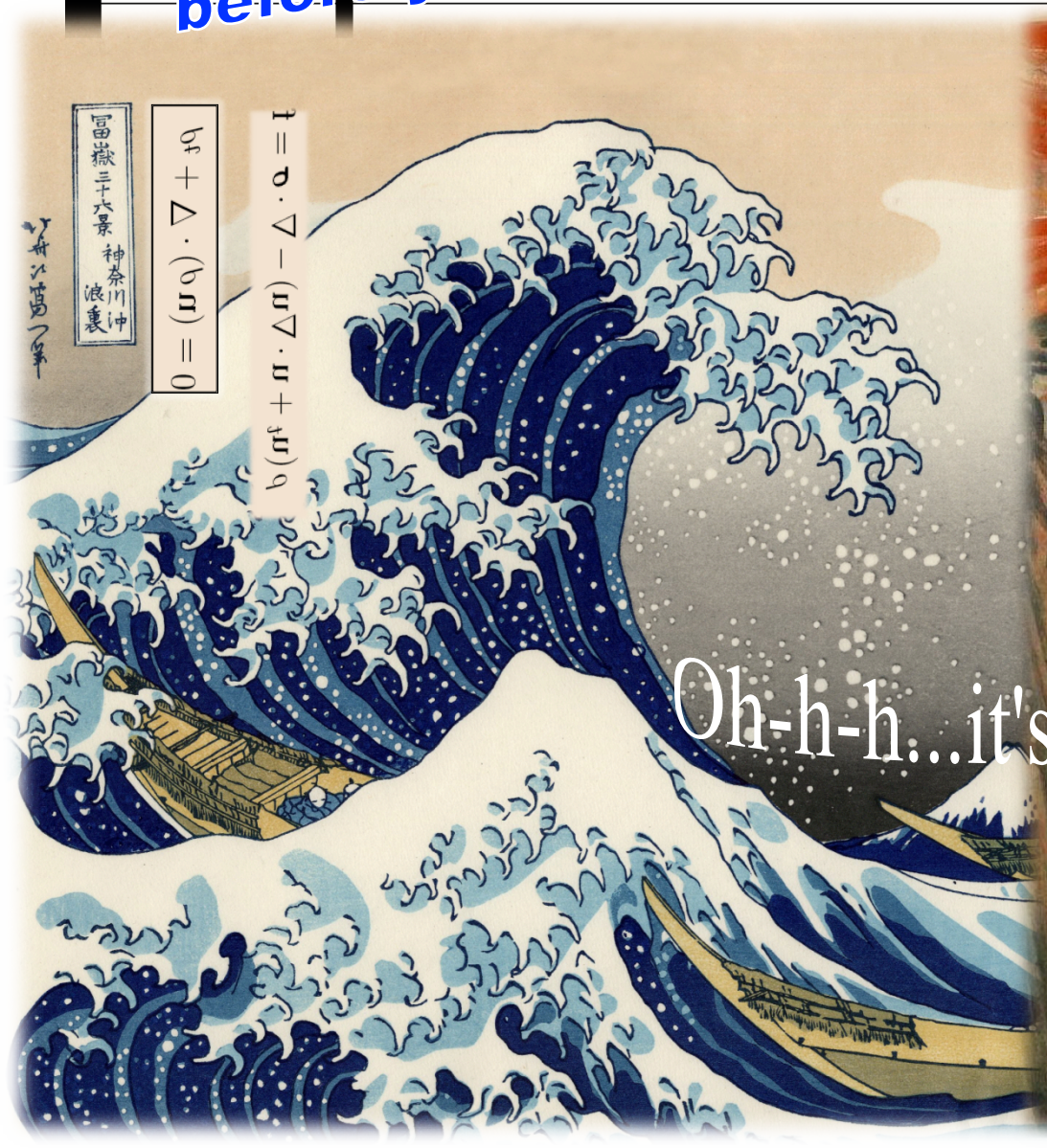
Università della Svizzera italiana

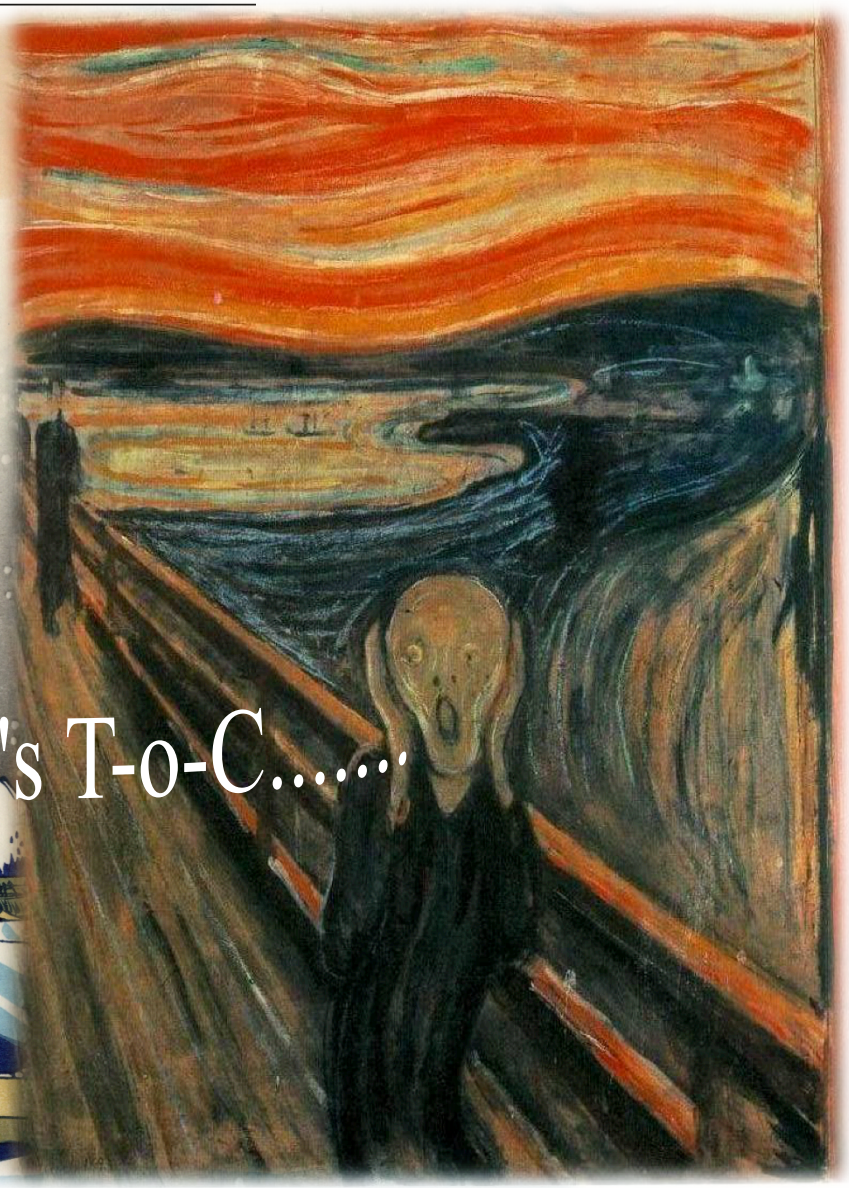
Facoltà di scienze informatiche

# All you need to know about CS before you graduate

## Intro to Theory of Computation

www. inf.usi.ch\faculty\sharygina



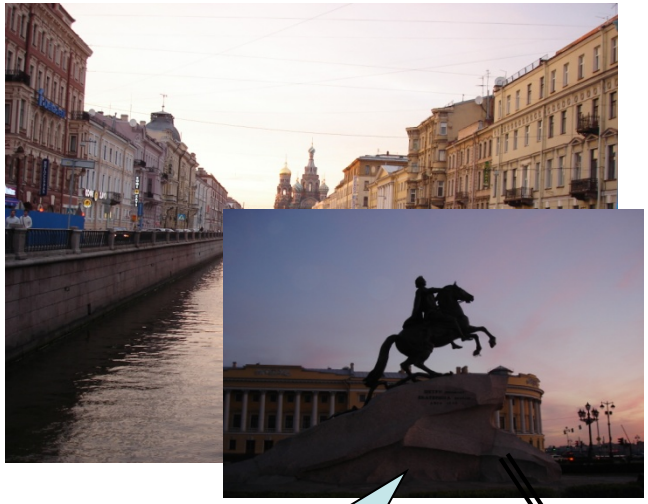



Università della Svizzera italiana

Facoltà di scienze informatiche

# About me

Professor, USI



Undergraduate and Graduate studies, Saint-Petersburg, Russia Politechnic Institute

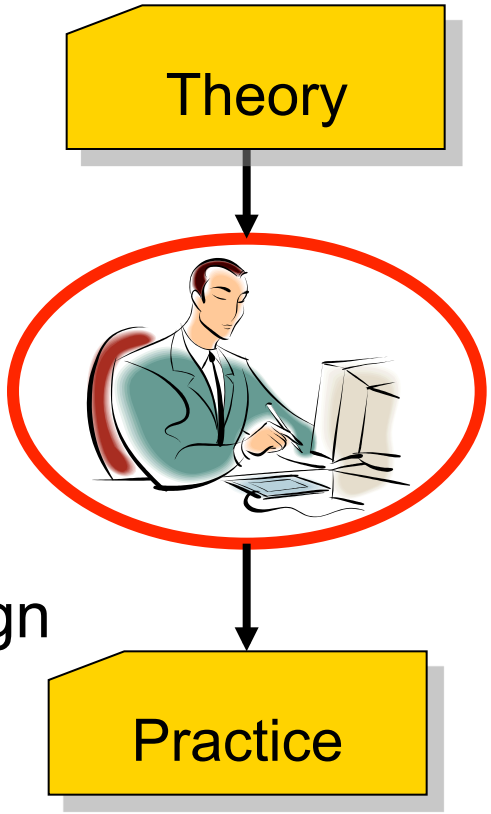
Ph.D.,The University of Texas at Austin, USA Research Faculty, Carnegie Mellon University, Pittsburgh, USA

Università della Svizzera italiana

Facoltà di scienze informatiche

# My Research

Computational Logic
Algorithms
Concurrency Theory




Example:
Bug detection
in software

Computer Aided Design
Program Analysis
Dependability

www.inf.usi.ch

# Class Introductions

- Martin Blicha, Ph.D. student at Formal Verification Group
- Pietro Benedusi, Ph.D. student
- Antti Hyvarinen, postdoctoral fellow
- and you …….

# My name is…

- First name
- Last name
- Nationality
- Other languages
- High school (where)
- Why informatics?
- Why USI?
- Your favorite course at USI?
- What do you think theory of computation is about?
- What is your strength?
- What is your dream job?
- What are your intentions after the graduation? (job, graduate degree, 2 years in Hawaii, etc.)

# Computing World

Let's try to generalize the world of computing?
(how it all started and where we are going...)

Let's put all your classes in perspective

How would you position yourself as a professional in this picture?

Are you confident about your expertise (can you solve any software design task)?

Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

# Theory of Computation

This course is about limits of computations

Università
della
Svizzera
italiana

Facoltà
di scienze
informatiche

# Theory of Computation

This course is about limits of computations

WHAT ARE THE BENEFITS TO US?

# Theory of Computation

**A 2-Question Course**:

- Computability Theory:

  What problems can(not) computers solve?

- Complexity Theory:

  What makes some problems computationally hard and others easy?

# Topics

## *Computability:*

*Turing machines (focus on expressiveness and applicability)*
*Decidability*
*Undecidability*
*Reducibility*

## *Complexity:*

*Time and space hierarchies*
*P and NP*
*NP-completeness and PSPACE-completeness*
*SAT and QBF problems: games, graph coloring, clique, etc. (as applications of the above)*

# Motivation

- **Relevance to practical applications**
  - Emagine you are a sys. admin at CIA. How do you asses stability of a network system or an individual running station with costly reloading? Why does it compute so long? Did it crash?
  - Why some encryptions are better to resist hacker attacks? How to design such stable encryptions?
  - Which problems can not be solved by programming?
  - What kind of problem is hard to solve quickly?

# Motivation (2)

- **Foundation for theoretical work**
  - Engine for new discoveries in computer science
  - Challenging open research problems (P vs. NP…)
- **It is fun**
- **...and money!**

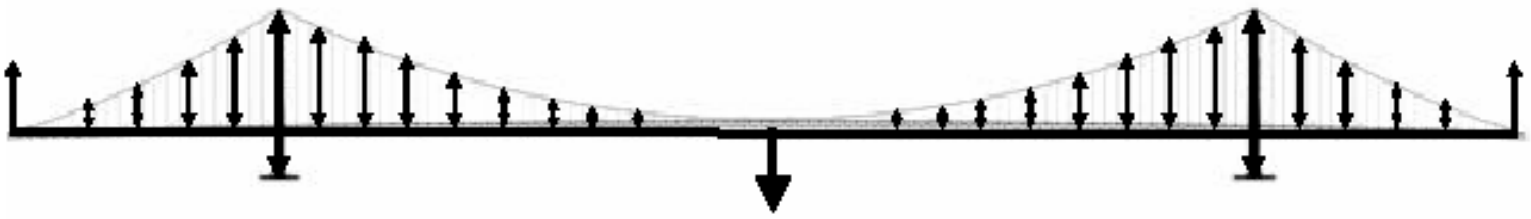
# Course Pre-requisites

- Both Theory of Computability and Complexity Theory deal with *Formal Models of Computation* *and* *Mathematical Logic*

  - Automata and Formal Languages class is required
  - Operational Knowledge of Set theory and Proof techniques (Data Structures and Algorithms class).
  - Basics of Complexity Measures (Big O Notation).

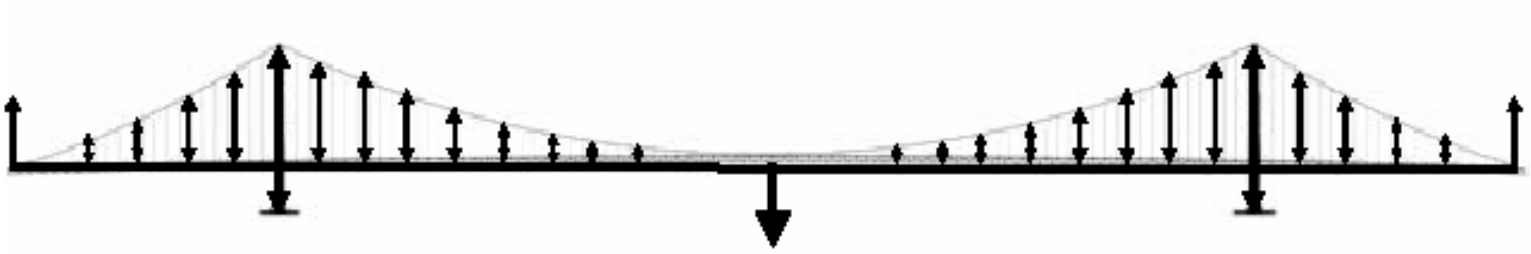
# Why build a model?

## Model: Free Body Diagram



# Why build a model?

## Model: Free Body Diagram



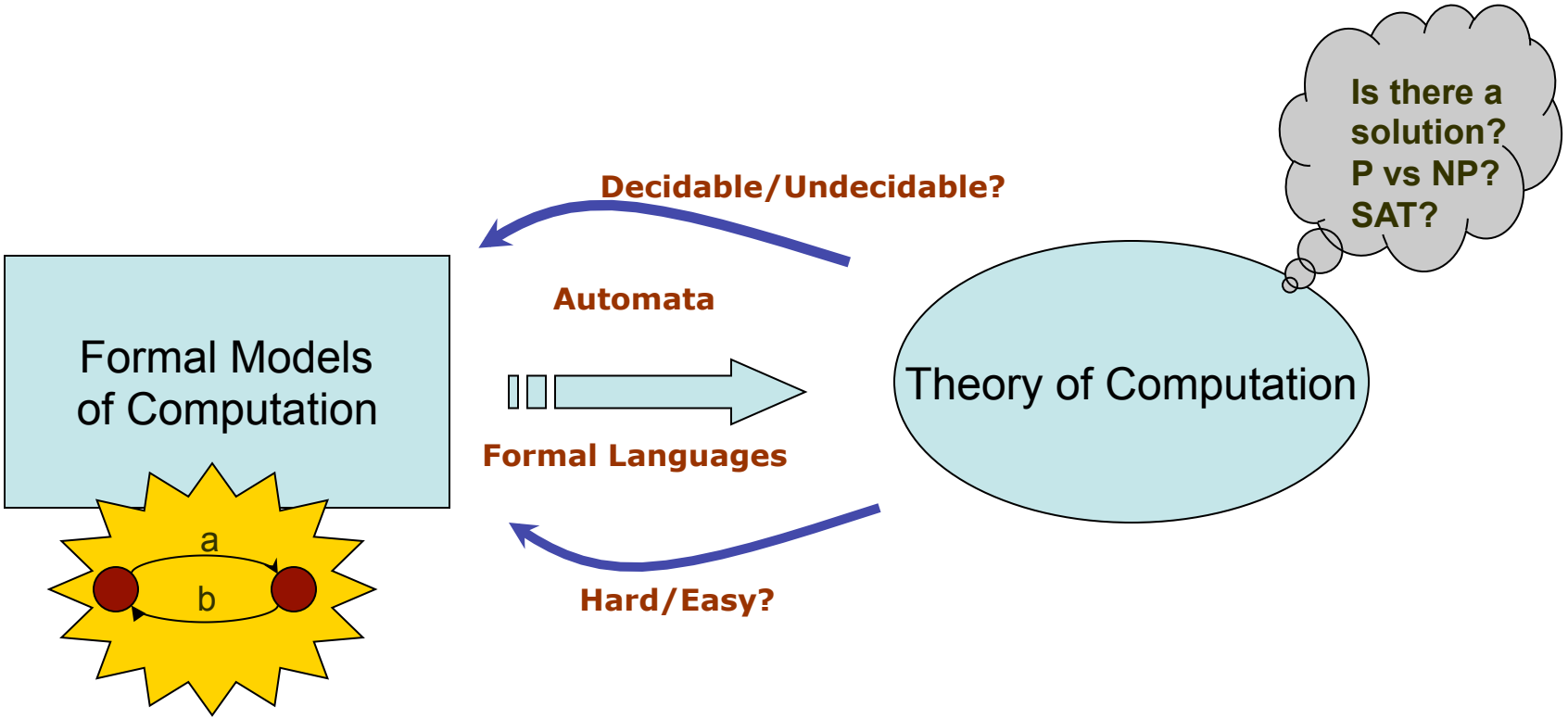
Analyzing a model helps to build reliable devices (e.g., bridges, etc)

In Computer Science we do the same with computers and computer programs

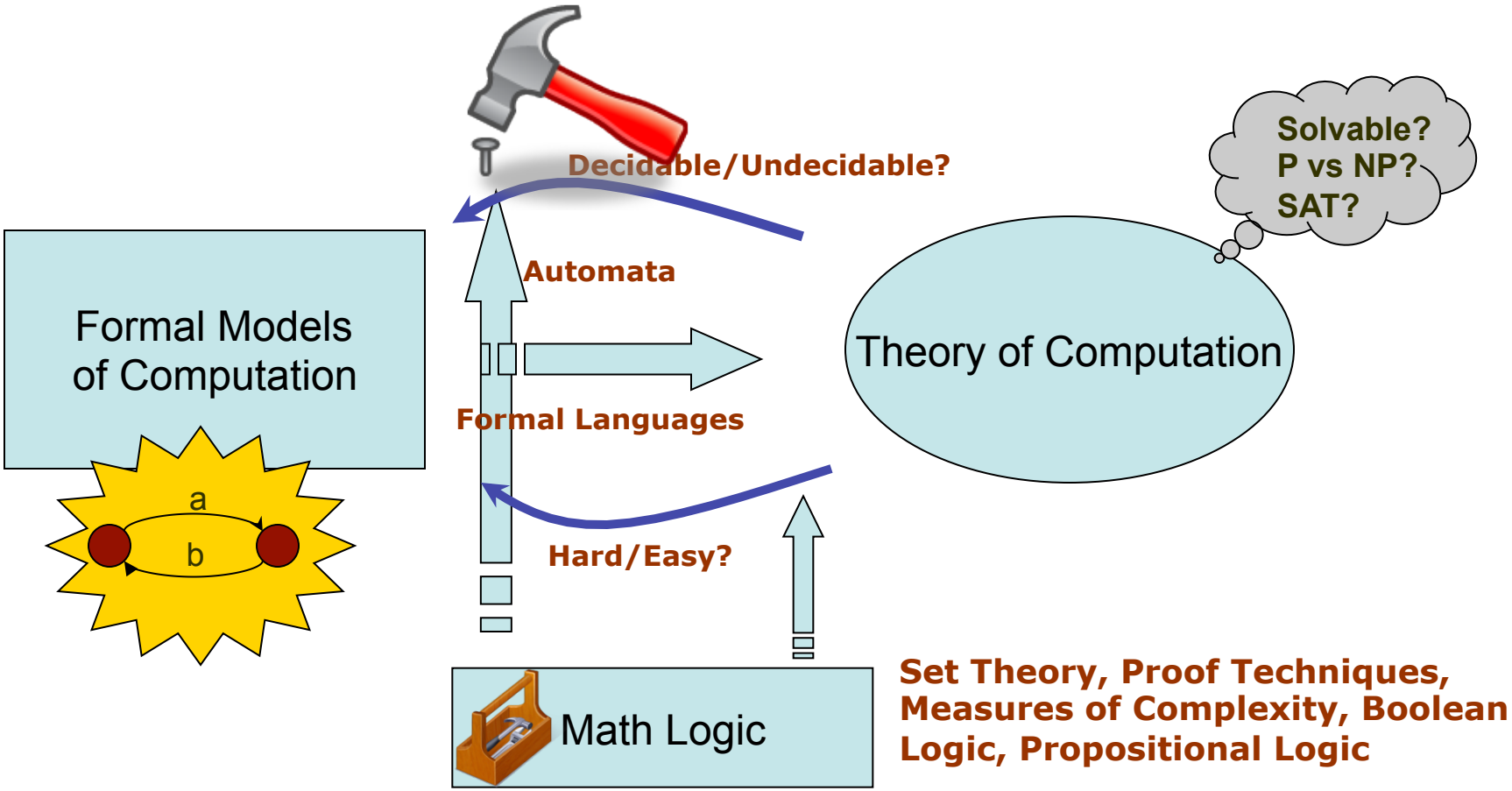
# Programs and their models

- Sequential programs can be modeled as pushdown automata
  - E.g., Microsoft Static Driver Verifier looks for bugs in device drivers after creating a pushdown automaton models for them
  - Etc… can you give any other examples?

# Course Pre-requisites



# Course Pre-requisites (2)



# Course Structure and Experience

- **Theory sessions**
  - new concepts and proofs
- **Lab sessions**
  - developing skills for solving practical problems
- **Homeworks and Exams** (exercizing theoretical concepts and proofs presented in the theory sessions)

# Course Structure

- **Evalutation**
  - HWs = 55% (theory and practice HWs)
  - Exam = 35%
  - Quizes = extra credits 2% each
  - Participation = 10%
  - Bonus assignements
- **Reading assignements (Important!)**
- **Individual work on homeworks**

# Course Organization

- **Moodle**
  - **Title: Theory of Computation**

# Course Organization

- Reading assignments
  - Located at the date when they are due!
- Homework assignments
- Announcements
  - For you, to ask (and answer) questions
- Discussion forum
  - From us (e.g. about schedule changes)
- Slides and other additional resources

# Workload – 6 ECTS

- **1/5** of your workload this semester (of 30)
- 150 – 180 working hours total (of 750-900)
  - Class sessions: 22 * 90 minutes = 33 hours
  - Self-study: **117 – 147 hours (78%-81%)**
- 11 – 13 working hours per week (14 weeks)
  - Class sessions: 3 hours/week
  - Self-study: **8 – 10 hours/week**

# Collaboration Policy

- We encourage collaborative study
  - You **should** work together to study assigned reading
  - You **should** work together to solve homework assignments
- However, we do not tolerate plagiarism/cheating
  - You **have to** give explicit credit to everyone who collaborated with you or who helped you, and to all web sites, articles, or books you used
  - You will be on your own in midterm and final exams

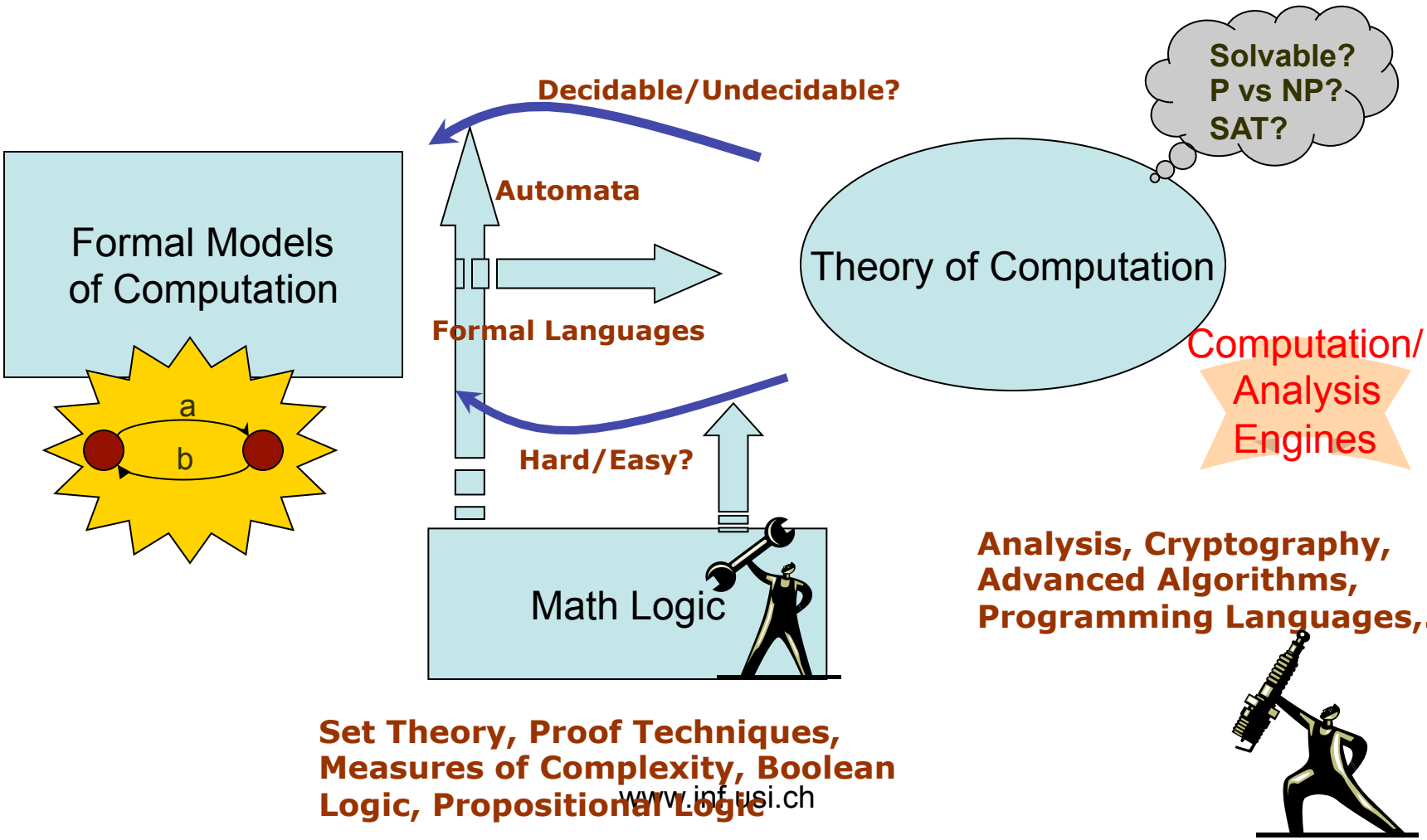
# Skills

- Thorough understanding of several models of computation, their purpose, and their use in formal arguments.

- Exposure to fundamental results in complexity theory and building of intuition regarding limitations of computing.

- Skills at manipulating formal systems as used in CS theory, particularly in developing and writing up proofs.

# Course Relation to Grad. Studies



# Observations about the subject

- This is a *simple* class ((fairly few) concise and well-defined concepts and techniques are introduced)

- This is a *difficult* class (it requires logical decision and often relies on applying various techniques from different disciplines. Often there are multiple solutions to the same problem).

- It is a good match as a concluding course of our bachelor's program: develops problem solving skills, puts together a lot of concepts from various subjects, prepares for graduate studies, and life in general!