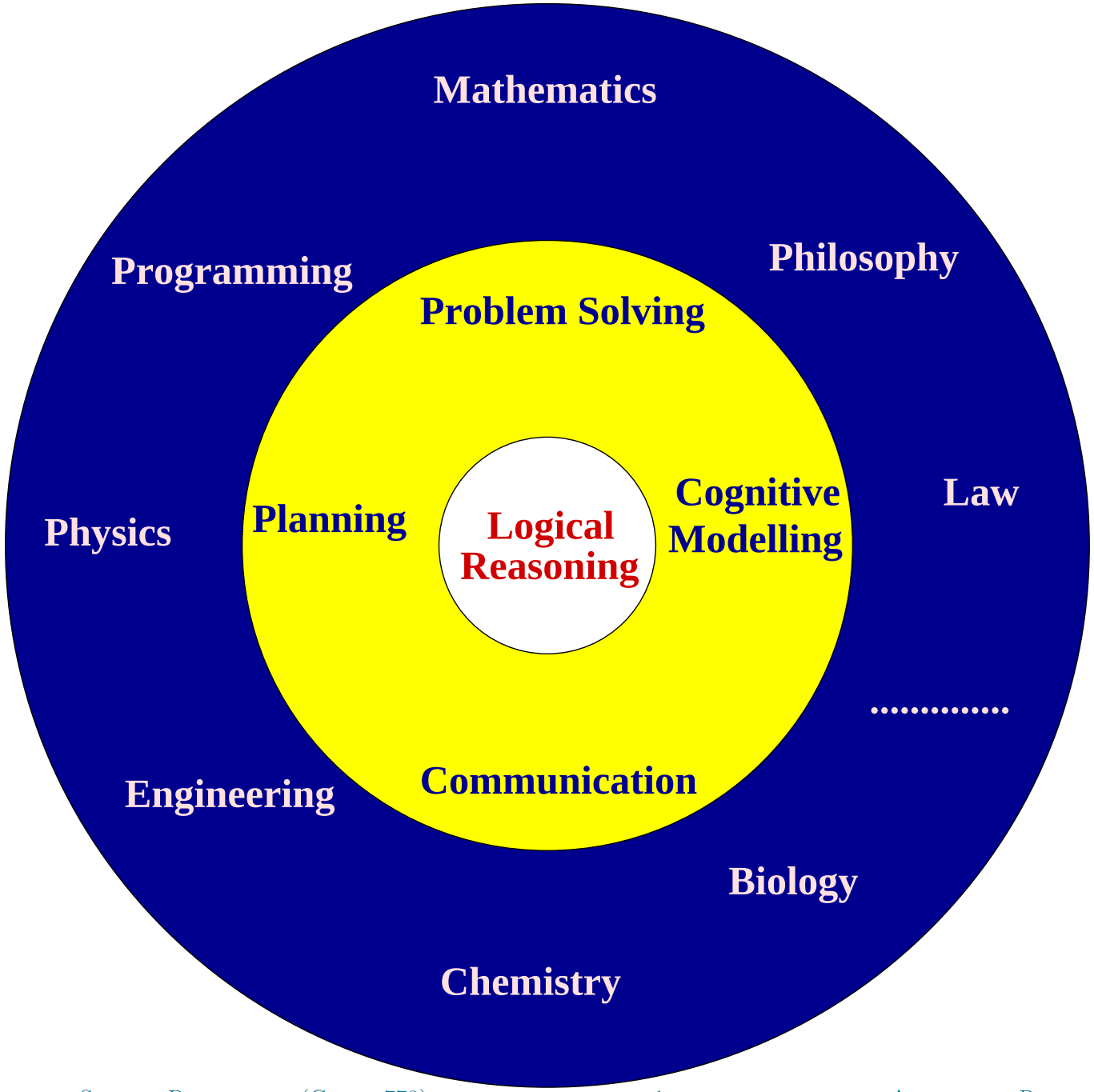
Automated Reasoning

Christoph Kreitz

1. **What is Automated Reasoning?**
2. **Why Automated Reasoning?**
3. **Achievements**

Logical Reasoning – the Center of Intelligence



Automated Reasoning: Leibniz's dream fulfilled ?

∼1700: "Make logical reasoning precise"

– A universal & accurate scientific language
– Rules for evaluating scientific arguments

$\Big\}$ $\mapsto$ *"Calculemus"*

1890: Formal logics

– Formal language + Inference rules
– Laws of though expressed by mechanical manipulation of text

1950: Computers – tools for symbolic manipulations

– Error-free application of rules
– Ability to search for solutions by exploring millions of possibilites

$\Downarrow$

Simulate logical reasoning on a computer

Major Application Areas

- **Prove mathematical theorems**
  - Detect and correct errors in proofs (Proof Checking)
  - Find new proofs automatically (Theorem Proving)
- **Support development of reliable software**
  - Find bugs / prove correctness (Verification)
  - Improve performance (Optimization)
  - Generate from specifications (Synthesis)
- **Inference engine for AI-Systems**
  - General problem solver, robot planning, ...

Proof Calculi: Foundation of Automated Reasoning

- **Formal language**
  - Syntax: expressions built from parameters and logical symbols
  - Semantics: logical symbols have fixed meaning
- **Proof calculus**
  - Inference rules for symbolic manipulation of expressions
  - Must be proven correct and complete wrt. semantics

$$\frac{A \quad A \Rightarrow B}{B}$$

- **Many logics for different purposes**
  - Classical logic (standard mathemantics)
  - Modal logics (knowledge and belief)
  - Constructive / intuitionistic logic (methods and programming)
  - Linear logic (resources, actions, planning, ...)
  - Nonmonotonic logics, default logic, probabilistic logic, ...

    ⋮
  - Type Theory: higher-order logic + programming language + data types

Theoretical Limitations

- **There are no general algorithms to decide**
  - whether a given logical formula is valid
  - whether a given program terminates
  - whether a given program is correct
  - whether two programs have the same functionality
- **We can only search for positive results**
  - Infinite search tree – no answer in negative case
  - Search techniques from AI do not apply to Theorem Proving, Software Verification, or Program Synthesis

$$\Downarrow$$

Intelligent proof techniques required

Machine Support for Formal Reasoning

- **Interactive Proof Editors**
  - User constructs proofs interactively by applying rules
  - Machine executes rules and returns unsolved subproblems
  - Basic mechanism: pattern matching + term rewriting
- **Automated Proof Procedures**
  - Tactics: programmed application of individual reasoning steps
  - Decision Procedures for restricted domains
  - Proof Search strategies, complete for small logics
    - Resolution, Matrix Methods, Model Checking, ...
  - Knowledge-Based Reasoning: search guided by domain knowledge

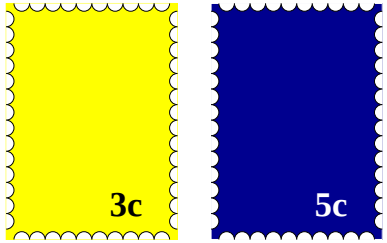
Systems for Automated Reasoning

- **Proof Development Systems** $\mapsto$ Build Formal Knowledge
  - · NuPRL, PVS, HOL, Coq, KIV, Ωmega, ...
  - – Human user guides proof system
  - – Proof editors enhanced by tactics and/or decision procedures
  - – Additional support: libraries, definitions, program evaluation, ...
- **Automated Theorem Provers** $\mapsto$ Find Proofs
  - · Otter/EQP, Setheo, Gandalf, Spass, ...
  - – No interaction
  - – Search + Unification, Paramodulation, AC-Matching, Rewriting ...
  - – Search parameters can be modified, lemmas can be provided ...
- **Special Purpose Systems** $\mapsto$ Synthesize Software
  - · KIDS, SpecWare, VSE, ...
  - – System guides human user through choice points
  - – Search strategies taylored for application domain

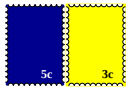
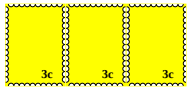
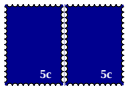
Why Automated Reasoning?

- **Too many errors in informal reasoning**
  - 40–50% of the results published in journals are wrong ↦ Formalization
- **Software controls major aspects of our life**
  - Air traffic, Banking, Government, Utilities, Schools, ...
  - Errors can be annoying (Reboot, loss of data, ...)
  - Errors can be very expensive (Pentium bug, failed rocket launches)
  - Errors can cost lives (Airbus crashes in the early 1990's)
- **Software Development unreliable**
  - Tested programs still contain errors
  - Correctness proofs are tedious and error-prone ↦ Automatization

An example from Mathematics: the Stamps Problem



Can you represent any postage of 8 cents or higher with only stamps of 5c and 3c ?

8c =  , **9c** =  , **10c** =  , **11c = 8c +**  **...** *YES*

- **Can you prove it?**
  - Precise formulation: $\forall$`n.` $\forall$`8` $\leq$ `x<8+3n.` $\exists$`i,j.x=5i+3j`
    - Inductive proof: we can represent `x+1` if we can represent `x-2`
    - Base cases: 8, 9, 10
- **Can you do the same with other pairs of stamps?**
  - Obviously 1c and any other stamp, 2c and any odd number
- **Can you prove that there are no others?**

If $\forall x \geq a+b.\ \exists i,j.\ x = i \cdot a + j \cdot b$ then `a=1` or `a=2, b` odd or `a=3, b=5` `(a<b)`

Assume `1<a<b` and do some number theory (if `a=1` we're done)

- $\exists i,j.\ a+b+1 = i \cdot a + j \cdot b \mapsto$ `a | (b+1)` or `b | (a+1)` (1.)
- $\exists i,j.\ a+b+2 = i \cdot a + j \cdot b \mapsto$ `a=2` or `a | (b+2)` or `b | (a+2)` (2.)
- $\exists i,j.\ a+b+3 = i \cdot a + j \cdot b \mapsto$ `a=3` or `a | (b+3)` or `b | (a+3)` (3.)

Case analysis

`a=2`**:** by (1.) `b` must be odd ✓

`a>2`**:** then `b>3`. Split into subcases according to (1.)

`a | (b+1)`**:** then `a` $\not|$ `(b+2)` and by (2.) `b | (a+2)` thus `b=a+2`

Split into subcases according to (3.)

- `b | (a+3)` is impossible since `b=a+2`
- `a | (b+3)` is impossible since `a | (b+1)` and `a>2`

Thus `a=3` and `b=5` ✓

`b | (a+1)`**:** then `b=a+1`

by (2.) `a | (a+3)` or `a+1 | (a+2)` both of which are impossible ✓

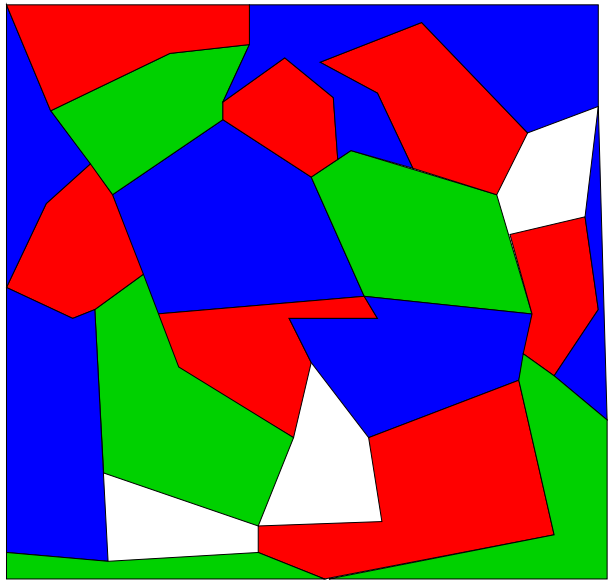
Problems of Informal Reasoning

- **If we're not forced to look at details we won't**
  - We usually trust the one who presents the proof
  - We care about the method, not the details
  - Our mental model may not capture all aspects
- **We jump to conclusions**
  - Proving only the complicated cases carefully
  - Inappropriate analogies
  - Hidden assumptions that are invalid in special cases
  - Ad håc solutions appear better than the are

Formal mathematical reasoning leads to deeper understanding and better solutions

Achievements: Four-Color Problem

Can every two-dimensional map be colored with only 4 colors?



- **Unsolved for several centuries**
- **Computerized proof in the late 70's**
  - Mathematicians reduced infinite number of situations to a few thousand cases
  - Computer verified that all cases can be colored with 4 colors
  - Special-purpose software, not really Automated Reasoning
  - Correctness of proof depends on (unverified) software

Achievements: Automated Theorem Proving

1995: **Proof of unsolved quasi group theorem**

1996: **Robbins Algorithm Conjecture** $\mapsto$ NY Times

- Open mathematical problem for more than 60 years
- Reduced to two sufficient conditions in 1980 (hand proof)
- EQP prover verified both conditions as sufficient in 1996
- EQP proved condition to be true in October 1996

Proof found automatically with general-purpose prover

- Depth-limit 70 $\mapsto$ 8 days, 49,548 equations generated and checked
- Resulting proof is sequence of 15 equations

1995: **Pentium Bug found by Model Checking**

- Division algorithm mapped incorrectly onto hardware tables
- Missing case detected as "countermodel"

$\mapsto$ No new hardware design without control by model checkers

Achievements: Program Synthesis (KIDS)

Synthesis of correct-by-construction algorithms

1990: **Efficient Costas-Arrays Algorithm**

1993: **US-Army Transport Scheduling Algorithm**

- General Program Synthesis Tool
- User chooses efficient algorithm structure + a posteriori optimizations
- Created within a few hours, correct-by-construction
- Generated LISP algorithm 2000 times faster than existing ADA program

$\mapsto$ Commercial production of scheduling algorithms with KIDS

Achievements: Proof Development Systems (NuPRL)

- **Mathematics**
  - Proof of Girard's paradox (1987)
  - Formal mathematical textbook
- **Hardware Design:**
  - Verification of logic synthesis tool (1993)
  - Verification of SCI Cache Coherency Protocol (1996)
- **Software Verification & Optimization:**
  - Code of Ensemble group communication system imported into NuPRL
  - Verification of total order protocol uncovered subtle bug (1998)
  - Fast-path optimization improved performance by factor 3 (1999)

Automated Reasoning vs. Human Intelligence

- **Formal inferences are "logical"**
- **Automated Reasoning shows intelligent behavior**
  - Proof tactics correspond to human reasoning strategies
  - Systems are personal reasoning assistants for experienced users
  - Systems can teach methodology to newcomers
- **Automated Reasoning is more accurate**
- **Automated Reasoning finds better solutions**
  - Can all brilliant ideas be found by search?

Will fully automated systems dominate the future of mathematics and programming or will we always need a human component ?