Computer-assisted formal reasoning — So what?

Markus Wenzel
TU Munich
May 2000

Overview
1. A bluffer’s guide to formal logic
2. Actual applications of logic
What is Logic anyway?

Syntax:
- concrete syntax: don't do it! (just apply parser technology)
- abstract syntax: use inductive types for example \( t \equiv x \mid f(t_1, \ldots, t_n) \) (the common way)
- higher-order abstract syntax: \( \lambda \)-calculus (very neat)

Inferences: define an inductive set of theorems
Example: \( P_1, \ldots, P_n \vdash P \) with rules
\[
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \quad \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}
\]

Semantics: by structural recursion on syntax
\[
M[P] = \ldots \text{ or } M \vDash P
\]
Example: Tarski-style “self-interpretation”
\[
M[A \rightarrow B] = M[A] \Rightarrow M[B]
\]

Note: Almost everything is a logic!
Theoretical studies

On syntax:
- type checking “\( t : \tau \)”
- type inference “\( t : ? \)”
- syntactic unification “\( t = ? u \)”

On inferences: “proof theory”
- derived and admissible rules
- proof conversions (“strong normalization”, “cut elimination”, etc.)
- complexity of derivations (“strength” of deductive systems etc.)

On semantics:
- useful: interpretation wrt. well-known systems (e.g. classical set-theory)
- general: “model theory” (categories)

On adequacy:
- correctness: \( \vdash P \implies \models P \)
- completeness: \( \models P \implies \vdash P \)
Example: Equational Logic

Syntax:
• first-order terms $t \equiv x \mid f(t_1, \ldots, t_n)$
• equations $P \equiv t_1 = t_2$

Inferences:
• equational reasoning:
  \[
  \begin{array}{c}
  x = y \\
  P\[x]\end{array} \Rightarrow \begin{array}{c}
  x = x \\
  P[\[x]\]
  \end{array}
  \]

• term rewriting:
  \[
  l \rightarrow r \Rightarrow \begin{array}{c}
  t[\sigma l] \rightarrow t[\sigma r]
  \end{array}
  \]

Semantics: “universal algebra”

Adequacy: Birkhoff’s Theorem
Example: Propositional Logic

Syntax:
propositions $P \equiv X \mid P_1 \rightarrow P_2$

Inferences:
\[
\frac{\vdots}{A \rightarrow B} \quad (\text{intro}) \quad \frac{A \rightarrow B \quad A}{B} \quad (\text{mp})
\]

Semantics:
- classical boolean algebra
  (admits axiom $A \lor \neg A$)
- $\lambda$-calculus with simple types:
  \[
  \begin{align*}
  \llbracket A \rightarrow B \rrbracket &= \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket \\
  \llbracket \text{intro} \rrbracket &= \lambda \quad \text{(abstraction)} \\
  \llbracket \text{mp} \rrbracket &= \cdot \quad \text{(application)}
  \end{align*}
\]
Example: Predicate Logic

Syntax:
- terms \( t \equiv x \mid f(t_1, \ldots, t_n) \)
- forms \( P \equiv t = t \mid P_1 \rightarrow P_2 \mid \forall x. P \)

Inferences:

\[
\frac{\quad}{x = x} \quad \frac{x = y}{P[x]}
\]

\[
\frac{[A] \quad \vdots}{\quad B} \quad \frac{A \rightarrow B}{A}
\]

\[
\frac{[x] \quad \vdots}{\quad A} \quad \frac{\forall x. A}{A[t/x]}
\]

Semantics:
- predicates in classical set-theory
- \(\lambda\)-calculus with dependent types:
  \[
  [A \rightarrow B] = \Pi x : [A].[B] \\
  [\forall x. A] = \Pi x : U.[A]_x
  \]
  (may generalize this idea to Type Theory)
Automated Reasoning (∼ 1960–1990)

The Universalist’s Approach: We build a system to solve any of your problems — While-U-Wait!

Based on Classical First-order Logic (FOL) — why?

Theoretical virtues:
+ may express all of standard mathematics
+ complete inference systems available
+ amenable to fully automated proof search

Practical problems:
− expressiveness too weak
  (no types, no higher-order features)
− completeness mostly irrelevant
  (would rather prefer nicer logic)
− fully automated search unmanageable
  (too complex, fails ungracefully)

Options:
1. weaken the logic, use special automated tools
2. strengthen the logic, improve the working environment
Actual applications of logic

Aims:
• falsification: exhibit errors systematically
• verification: produce actual theorems
• proving: actual theorems and explicit proofs

Approaches:
• specialized push-button systems for generic users
• generic environments for specialized users

Example products:
• Model-checking tools: Spin, SMV, ...
• PVS: “Combining specification, proof checking, and model checking” (many bugs found so far)
• hol98: “An industrial prover” (long tradition of hardware verification)
• Isabelle: “A generic theorem proving environment” (the generic-everything approach)
• Coq: “A proof assistant for type theory” (the French way)
Some special theories

**Decidable:**
- propositional logic
- propositional temporal logic
- restricted fragments of FOL, e.g. $\forall^*\exists^\forall^*$
- linear integer arithmetic (Presburger): FOL over $(\mathbb{Z}, <, =, +)$
- real algebra (Tarski): FOL over $(\mathbb{R}, <, =, +, \times)$
- monadic second-order logic ($\rightarrow$ Mona)
- unification of “higher-order patterns” (!)

**Undecidable:**
- integer arithmetic (Gödel): FOL over $(\mathbb{N}, +, \times)$
- general second-order logic
- general higher-order unification (!)
  (equational theory of $\beta,\eta$ on $\lambda$-terms)
Example: Temporal Logic

Syntax:
propositions \( P \equiv X \mid P \to P \mid \Box P \)

Inferences:

\[
\begin{array}{c}
\vdots \\
B \\
A \to B \\
A \\
\Box A \\
\Box (A \to B) \\
\Box B
\end{array}
\]

additional structural axioms:
\( \Box A \to A \) (refl), \( \Box \Box A \to \Box A \) (trans), etc.

Semantics: Kripke structures

Applications:
- modeling behavior of distributed systems
- falsification ("verification") via Model-checking
Example: Higher-Order Logic (HOL)

Syntax:
- terms $t \equiv x \mid t_1 \ t_2 \mid \lambda x.t$
- types $\tau \equiv \alpha \mid \tau_1 \rightarrow \tau_2$

Inferences:

\[
\begin{array}{c}
\frac{}{A \rightarrow B} \\
[\alpha]
\end{array}
\quad
\begin{array}{c}
\frac{A \rightarrow B}{A[B]} \\
\frac{}{A}
\end{array}
\quad
\begin{array}{c}
\frac{}{\forall x. A} \\
[\alpha]
\end{array}
\quad
\begin{array}{c}
\frac{\forall x. A}{A[t/x]}
\end{array}
\]

almost everything is a defined concept

Semantics: simply-typed classical set-theory
($\approx$ plain mathematics)

Characteristics:
- simple
- powerful
- versatile
- well-balanced: pure vs. applied logic
Isabelle/HOL

Standard HOL theories:

Standard HOL packages:
Applications of HOL:

- abstract representation of logics ("logical framework")
  e.g. FOL, HOL, ZF, CC within minimal HOL
- semantic representation of logics ("shallow embedding")
  e.g. temporal logic, Hoare logic in classical HOL
- classical mathematical modelling
- meta-theory of systems and languages
  e.g. Java semantics
- eProof (?)
- . . .

http://isabelle.in.tum.de
http://isabelle.in.tum.de/Isar