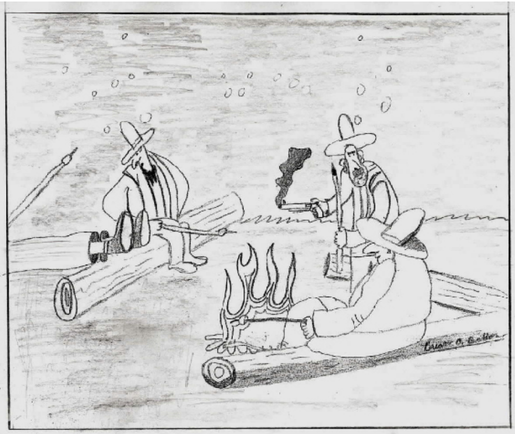
Computer Aided Formal Reasoning (G53CFR, G54CFR)



You guys are both my witnesses... He insinuated that ZFC set theory is superior to Type Theory!

Zermelo-Fraenkel Set Theory

Zermelo (1871-1953) Fraenkel (1891-1965)

- Axiomatic Set Theory $\approx$ 1925
- ZFC = Zermelo-Fraenkel with Axiom of Choice
- Foundations of modern Mathematics
- Additional axioms, e.g. the continuum hypothesis

Axiom of extensionality $\forall x \forall y [\forall z (z \in x \Leftrightarrow z \in y) \Rightarrow x = y]$

Axiom of regularity $\forall x [\exists a (a \in x) \Rightarrow \exists y (y \in x \land \neg \exists z (z \in y \land z \in x))]$

Axiom schema of specification $\forall z \forall w_1 \ldots w_n \exists y \forall x [x \in y \Leftrightarrow (x \in z \land \phi)]$

Axiom of pairing $\forall x \forall y \exists z (x \in z \land y \in z)$

Axiom of union $\forall \mathcal{F} \exists A \forall Y \forall x (x \in Y \land Y \in \mathcal{F} \Rightarrow x \in A)$

Axiom schema of replacement $\ldots$

Axiom of infinity $\ldots$

Axiom of power set $\ldots$

Axiom of Choice $\ldots$

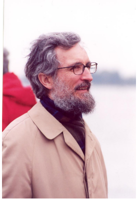
Set Theory for Computer Science?

- Set Theory is untyped (everything is a set), while programming languages are typed (either statically or dynamically).
- Basic concepts from computer science (records, functions) are not primitive in Set Theory.
- Basic operations in set theory (e.g. $\cap$, $\cup$) are not directly available on types.
- Set Theory is not constructive, i.e. there is a set theoretic *function* solving the Halting Problem.

Question:

Is there an alternative to Set Theory?

Martin-Löf Type Theory



Per Martin-Löf (1942-)

- Martin-Löf introduced Type Theory as a *constructive foundation of Mathematics* since 1972.
- Type Theory doesn't rely on predicate logic but uses types to represent propositions.
- Basic operations on types are Π-types (dependent function types) and Σ-types (dependent records).
- Type Theory is a programming language.

Propositions as types (The Curry-Howard Isomorphism)

- A proposition corresponds to the types of it proofs.
- A proposition is true if the corresponding type is non-empty.
- Conjunction $A \wedge B$ is repesented by cartesian product ($A \times B$).
- Implication $A \rightarrow B$ is represented by function types $A \rightarrow B$ (looks the same).
- $\forall$ and $\exists$ correspond to Π (depednent function) and Σ (dependent records).

Agda

Ulf Norell

- Ulf Norell has implemented Agda, a functional programming language based on Type Theory in his PhD in 2007.
- Agda is inspired by earlier systems such as Epigram, Cayenne and Coq.
- Agda can be used to program and to reason.

Course contents

1. Agda intro
2. Propositions as types (using Agda)
3. Dependently typed programming (in Agda)
   - Refining programs to certifiably correct programs
   - Representing data formats
   - Typed Domain Specific Libraries

Practicalities

- Two lectures: Tuesday and Thursday morning.
  The early student catches the first.
- Lab sessions each Friday 10:00, B52 (using Agda)
- Regular coursework (in Agda)
- Resources: available online
  `http://www.cs.nott.ac.uk/~txa/g53cfr/`

Assessment

G53CFR	40%	Exercises
	60%	Online exam
G54CFR	40%	Online exam
	60%	Project