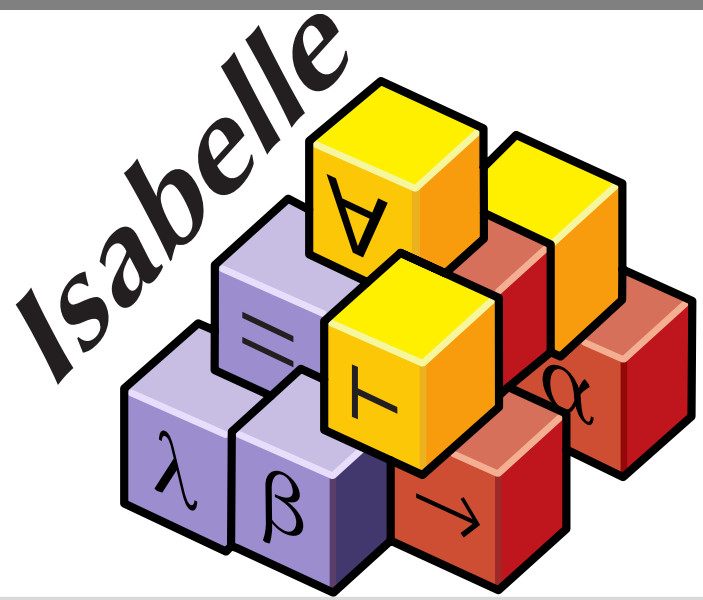
Karlsruher Institut für Technologie

Theorembeweiserpraktikum

Anwendungen in der Sprachtechnologie

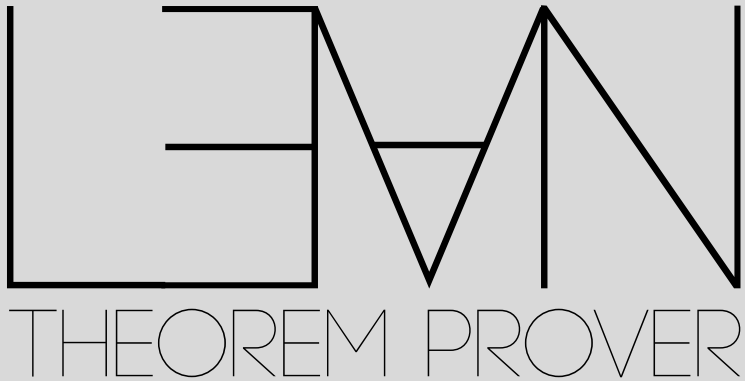
LEHRSTUHL PROGRAMMIERPARADIGMEN



Karlsruher Institut für Technologie

Teil XXXIII

Lean: Mit starken Typen abhängen



Demo: Aussagenlogik

Curry-Howard-Korrespondenz

Terme	$\leftrightarrow$	Beweise
Typen	$\leftrightarrow$	Aussagen

Curry-Howard-Korrespondenz

Terme	$\leftrightarrow$	Beweise
Typen	$\leftrightarrow$	Aussagen
Polymorphe Typen	$\leftrightarrow$	Aussagenlogik zweiter Ordnung
Abhängige Typen!	$\leftrightarrow$	Prädikatenlogik
$\Pi x : A.B$		$\forall x : A.B$

Curry-Howard-Korrespondenz

Terme	$\leftrightarrow$	Beweise
Typen	$\leftrightarrow$	Aussagen
Polymorphe Typen	$\leftrightarrow$	Aussagenlogik zweiter Ordnung
Abhängige Typen!	$\leftrightarrow$	Prädikatenlogik
$\Pi x : A.B$		$\forall x : A.B$

$\Rightarrow$ Eine Sprache, ein Typsystem sowohl für Definitionen als auch Beweise

Pi-Typen

- Einfach getypter Lambda-Kalkül

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t \ : \ \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2}$$

Pi-Typen

- Einfach getypter Lambda-Kalkül

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t \ : \ \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2}$$

- Calculus of Constructions *[Coquand, Huet 1986]*

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t \ : \ \Pi x : \tau_1 . \tau_2}$$

$$\frac{\Gamma \vdash e_1 \ : \ \Pi \ x : \tau_1 . \ \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2 [e_2 / x]}$$

Pi-Typen

- Einfach getypter Lambda-Kalkül

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t \ : \ \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2}$$

- Calculus of Constructions *[Coquand, Huet 1986]*

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t \ : \ \Pi x : \tau_1 . \tau_2}$$

$$\frac{\Gamma \vdash e_1 \ : \ \Pi \, x : \tau_1 . \, \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 \ e_2 : \tau_2 [e_2 / x]}$$

$$\tau_1 \to \tau_2 \equiv \Pi \ \_ : \tau_1 . \, \tau_2$$

Demo: Prädikatenlogik

Syntaktischer Vergleich

Isabelle/HOL	Lean
$\Rightarrow$ $\Longrightarrow$ $\longrightarrow$ $\bigwedge$ $\forall$	Π

`f x`
`g[of x]`
`h[OF g]`
	`f x`
	`g x`
	`h g`

`record`, `locale`, `class` `datatype`, `predicate`	`structure` `inductive`

Typen gut, alles gut?

Ein stärkeres Typsystem erschwert leider auch die Implementierung von Automation. Lean hat bisher

- ein grundsätzliches `simp`, “conditional term rewriter”
- SMT-Primitive wie E-Matching und Congruence Closure
- kein `auto`, ...
- keinen Sledgehammer

Typen gut, alles gut?

Isabelle	Lean
erstes Release 1986	erstes Release 2015
680.000 LOC (HOL) + 2.100.000 LOC (AFP)	28.000 LOC + 68.000 LOC (mathlib)

Lean 3

- Benutze Lean als seine eigene Metaprogrammiersprache
  - Definiere Taktiken, Syntaxerweiterungen, Debugger, . . . in Lean
  - Strikte Trennung zu konsistentem Sprachteil
- Führe Metaprogramme in (relativ effizienter) VM aus
- Baue auf effizienten in C++ definierten Primitiven auf
- *A Metaprogramming Framework for Formal Verification*
  Ebner, Ullrich, Roesch, Avigad, de Moura; ICFP 2017

Lean 4

- Teilweiser Rewrite begonnen 2018
- Interne Vereinfachungen und Refactorings
- Neuer Parser implementiert in Lean
  - Erlaubt erweitertes Backtracking
  - Baut einen verlustfreien konkreten Syntaxbaum auf

  $\Rightarrow$ Basis für Refactorings, Editorunterstützung, Dokumentenerzeugung, ...
  - Lisp/Racket-artiges Makrosystem für hygienische Abkürzungen
- Neues Backend implementiert in Lean
  - Anbindung an LLVM für JIT- und native Kompilierung

  $\Rightarrow$ Lean als allgemeine Programmiersprache

Lean im Netz

`https://leanprover.github.io`

Theorem Proving in Lean – mit Online-Editor
`https://leanprover.github.io/theorem_proving_in_lean/`