Coqのふんいき

論理学友の会 #14

yak_ex

自己紹介

- 新 康孝(あたらし やすたか)
- Twitter: yak_ex / GitHub: yak1ex
  - 発表資料のソースはGitHubに上げています
- 愛知県在住、仕事では車載組込のべたC言語
- Software Foundations自習中のCoq初学者

Disclaimer

- 筆者はCoq初学者です。関数型言語に強いわけでもありません。
- 理論的背景等は流して「ふんいき」だけお楽しみください。
  - 「Coqという名前を聞いたことしかない」くらいが対象レベルです。
  - 今回は量化子、Inductive、inductionあたりも紹介しません。
- 用語の扱いや、特に用語の読み方は適当度が高いです。

Agenda

- Coqとは？
- Coqの(著名な)実績
- Coqの基盤
- Coqの論理体系
- インストール
- 使ってみる
  - 二重否定除去、排中律、パースの法則etcの同値性を示してみる

Coqとは？

- Proof assistant
  - 定理証明支援系
    - Interactive theorem prover
      - 自動証明ではなくてあくまでも人が証明する支援
- フランス国立情報学自動制御研究所(INRIA)開発
  - 関数型言語OCamlを開発、CoqもOCaml製
  - オープンソース(LGPL v2.1)

Coqの(著名な)実績

- 定理証明支援系として
  - 四色定理(Gonthier, Georges (2005))
  - Feit-Thompson Theorem(Gonthier, Georges (2012))
- (依存型の使える)プログラミング言語として
  - CompCert(Cコンパイラ)

CompCert

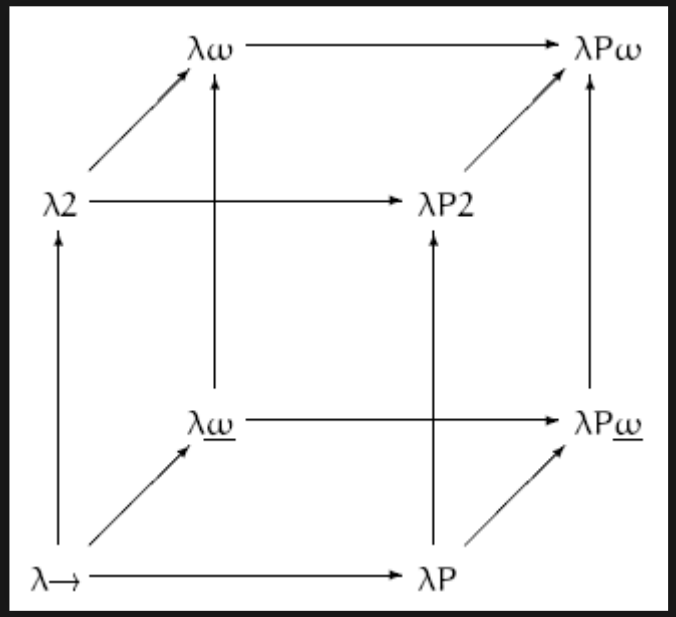
Semantic preservation theorem:

For all source programs S and compiler-generated code C, if the compiler, applied to the source S, produces the code C, without reporting a compile-time error, then the observable behavior of C improves on one of the allowed observable behaviors of S.

https://compcert.org/man/manual001.html#sec3

“improve”: runtime errorの置き換え

Coqの基盤

- Calculus of Inductive Consturction
  - 型付きラムダ計算の全部入り＋Inductive型
    - λ2(多相型：型に依存した値)
    - λω(型演算：型に依存した型)
    - λP(依存型：値に依存した型)
    - 

Coqの論理体系

- 直観主義論理
  - 二重否定除去等がデフォルトだとない
  - 必要なら自分で公理(Axiom)を入れてやればよい
- 関数外延性(functional extensionality: $\forall x, f x=g x \implies f = g$)がデフォルトだとない
  - 必要なら自分で公理(Axiom)を入れてやればよい

インストール

- opam使ってOCamlと合わせてインストールがおすすめ
  - WindowsだとWSL+opam+VS Code+Remote-WSL+VSCoqでOK

```
sudo apt install make m4 gcc unzip
sh <(curl -sL https://raw.githubusercontent.com/\
ocaml/opam/master/shell/install.sh)
opam init --disable-sandboxing # sandboxing無効はWSL
eval $(opam env)
opam install opam-depext
opam depext coq
opam install coq
```

使ってみる前に1

- 拡張子は.v
- VSCoqで左右レイアウトだと左側で編集、実行、右上が仮定、右下が帰結(ゴール)
- Alt+↑、Alt+↓で実行ポイントを上下
  - Alt+→でカーソル位置まで実行
- LemmaもTheoremもCorollaryもみんな一緒
- 今回は古典論理を古典論理たらしめている命題(二重否定除去等)の同値性を示してみる

使って見る前に2

- Coqの証明はtacticを適用していくことで実施(するのが普通)
  - intro/intros: ゴールから仮定に持ってくる
  - unfold: 定義を紐解く
  - destruct: 色々(仮定の分解系)
  - left/right/split: ゴールの分解
  - apply: 仮定、 定理等の適用
  - assumption: 仮定
  - exfalso: 爆発律
- 素のtacticは整理されていない→SSReflect

SSReflect

- 元々は拡張(四色定理の証明が契機)
- Coq/SSReflect/MathCompによる定理証明:フリーソフトではじめる数学の形式化
  https://www.amazon.co.jp/dp/4627062419

- coqtokyo Ssreflect勉強会 (名大講義の輪読会)

実演

https://github.com/yak1ex/imprecise_introduction_of_coq/blob/master/classical.v

終わりに

Software Foundations

やろうぜ！