STTwU in Coq

Freek Wiedijk

Radboud Universiteit, Nijmegen

Free University, Amsterdam

TYPES 2004

2004 12 15, 15:10

overview

the three points of this talk

- advertisement for Farmer's STTwU system
- report on a small Coq formalization
- how to map classical mathematics into constructive mathematics

isn't that **just** the double negation translation?

no:

extensionality of equality?

choice?

classical higher order logic in Coq

HOL

simply typed lambda calculus + ML-style polymorphism

type constants: `bool`, `ind`, $\rightarrow$

term constants: $=$, ε

$$\sqrt[n]{x} := \varepsilon y.\,(y^n = x \wedge (0 < x \Rightarrow 0 < y))$$

importing into Coq: two axioms needed

classical logic	`forall A:Prop, not (not A) -> A`
choice operator	`forall A:Set, not (notT A) -> A`
extensionality	`forall (A B:Set) (f g:A -> B),` `(forall x:A, f x = g x) -> f = g`

STTwU

Bill Farmer IMPS system $\longrightarrow$ LUTINS logic

Logic of Undefined Terms for Inference in a Natural Style

Simple Type Theory with Undefinedness

simply typed lambda calculus + undefined terms

type constants: `bool`, `ind`, $\longrightarrow$

term constants: $=$, ι

choice operator becomes partial

$$(\iota x.\, Px)\!\downarrow \;\Leftrightarrow\; \exists! x.\, Px$$

`bool` is special: all terms of type `bool` are defined

the STTwU system as an LF context

```
Parameter _type : Type.
Parameter _expr : _type -> Type.

Parameter _bool : _type.
Parameter _ind : _type.
Parameter _fun : _type -> _type -> _type.

Parameter _infer : _expr _bool -> Type.

Parameter _app : forall alpha beta:_type,
  _expr (_fun alpha beta) -> (_expr alpha -> _expr beta).
Parameter _lambda : forall alpha beta:_type,
  (_expr alpha -> _expr beta) -> _expr (_fun alpha beta).

Parameter _eq : forall alpha:_type,
  _expr alpha -> (_expr alpha -> _expr _bool).
Parameter _iota : forall alpha:_type,
  (_expr alpha -> _expr _bool) -> _expr alpha.
```

Coq formalization

a constructive model for STTwU in Coq

classical logic	double negation translation
extensionality	**setoids**
partial logic	**partial** setoids
choice operator	**dual** of partial setoid

no axioms needed!

a way to do the double negation translation

classical propositions $=$ copy of the **constructive propositions**

```
Definition __bool := Prop.
```

... but interpreted **negatively**

```
Definition __true : __bool := False.
Definition __false : __bool := True.

Definition __implies (A B : __bool) : __bool := ~(B -> A).

Definition _infer (A : __bool) : Type := ~A.

Definition __not (A : __bool) : __bool := __implies A __false.

Lemma __notnot_elim:
  forall A : __bool, _infer (__not (__not A)) -> _infer A.
```

the dual of a partial setoid

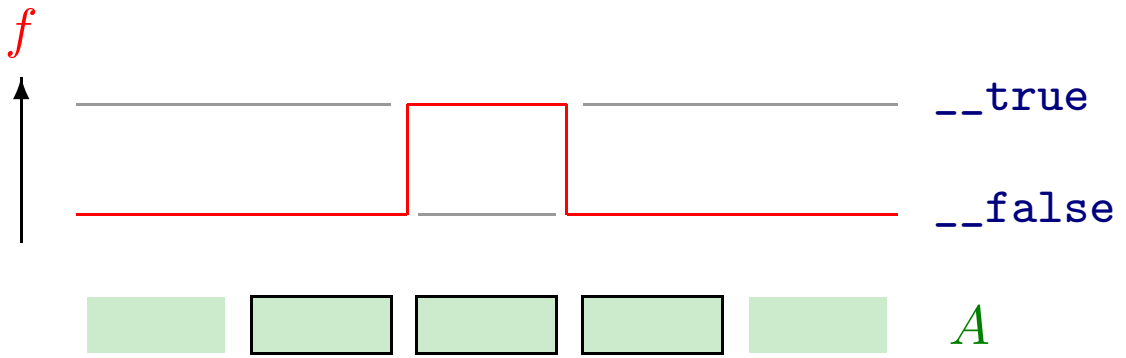
- **partial setoid**

  $(A, \sim)$ where $\sim$ is a **partial** equivalence relation on A

  a is 'proper element' when $a \sim a$

- **dual**

  partial setoid of functions $f : A \to$ `__bool`



$$f \sim g \;\Leftrightarrow\; \exists a.\, f(a) \wedge (\forall b.\, f(b) \Leftrightarrow a \sim b) \wedge (\forall b.\, g(b) \Leftrightarrow a \sim b)$$

$$\iota f.\, P(f) \;=\; \lambda a.\, (\exists f.\, P(f)) \wedge (\forall f.\, P(f) \Rightarrow (\forall b.\, f(b) \Leftrightarrow a \sim b))$$

. . . and can you really not do better than this?

Coq formalization heavily uses impredicativity of `Prop`

questions:

- is it possible to do the same in a predicative system?

  Agda

  NuPRL

- is it possible to do this for another Farmer system: STMM

  Set Theory for Mechanized Mathematics

  axiom NBG31 = the axiom of choice ?

  maybe work in: Gödel's constructible universe L