Formalizing Lindenbaum-Tarski algebra for propositional logic in Cubical Agda

Caroline Roos

Supervisor: Anders Mörtberg

Degree project

Department of Mathematics
Stockholm University

June 13, 2023

Formalizing mathematics

Why formalize?

Benefits of formalizing mathematics:

- Ensure correctness
- Detect errors in tradional mathematical proofs

There are many proof assistants out there
Coq, **Agda**, Lean, Idris, ...

Propositions as types

The propositions-as-types interpretation is the direct relationship between computer programs and mathematical proofs.

Prop	Type
$\top$	unit
$\bot$	void
$\phi_1 \wedge \phi_2$	$\tau_1 \times \tau_2$
$\phi_1 \supset \phi_2$	$\tau_1 \to \tau_2$
$\phi_1 \vee \phi_2$	$\tau_1 + \tau_2$

Becuase of strong typing and dependent types, Agda makes a good proof assistant.

Agda proof assistant

Defining a datatype in Agda

```
data Bool : Type where
  true : Bool
  false : Bool
```

Bool is the name of the datatype, and **true** and **false** are its constructors.

Agda proof assistant

Functions over datatypes can be defined using pattern matching.

$$\begin{aligned}&\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}\\ &\mathsf{not}\ \mathsf{true} = \mathsf{false}\\ &\mathsf{not}\ \mathsf{false} = \mathsf{true}\end{aligned}$$

The type of **not** is defined as a function from **Bool** to **Bool**. The function is then defined by pattern matching on the arguments.

Agda proof assistant

A **dependent type** is a type that depends on elements of another type.
For example, the polymorphic identity function:

$$\mathsf{id} : (A : \mathsf{Type}) \to A \to A$$
$$\mathsf{id}\ A\ x = x$$

In Agda it is possible to use implicit arguments.

$$\mathsf{id'} : \{A : \mathsf{Type}\} \to A \to A$$
$$\mathsf{id'}\ x = x$$

Agda will try to infer the type for us.

Cubical Agda

Cubical Agda is an extension of Agda that incorporates features of cubical type theory.

It has native support for set quotients!

Note: We are using the agda/cubical library, which is the standard library for Cubical Agda.

Formalizing propositional logic

#TODO

- Formulas and context?

Formalizing propositional logic

Choosing logical connectives

$$\top \quad \bot \quad \wedge \quad \vee \quad \neg$$

Formalizing propositional logic

Inference rules
All rules follow this general shape.

$$\begin{aligned}\wedge\text{-I} : \{\Gamma : \mathsf{ctxt}\}\ \{\phi\ \psi : \mathsf{Formula}\}\\ \to \Gamma \vdash \phi\\ \to \Gamma \vdash \psi\\ \to \Gamma \vdash \phi \wedge \psi\end{aligned}$$

$$\begin{aligned}\wedge\text{-E}_1 : \{\Gamma : \mathsf{ctxt}\}\ \{\phi\ \psi : \mathsf{Formula}\}\\ \to \Gamma \vdash \phi \wedge \psi\\ \to \Gamma \vdash \phi\end{aligned}$$

$$\begin{aligned}\wedge\text{-E}_2 : \{\Gamma : \mathsf{ctxt}\}\ \{\phi\ \psi : \mathsf{Formula}\}\\ \to \Gamma \vdash \phi \wedge \psi\\ \to \Gamma \vdash \psi\end{aligned}$$

Formalizing propositional logic

Law of excluded middle

The law of excluded middle states that for every proposition, either the proposition or its negation is true.

That is, for all formulas ϕ,

$$\vdash \phi \vee \neg\phi$$

This makes it a classical logic.
In Agda:

$$\begin{aligned}\mathsf{LEM} &: \{\phi : \mathsf{Formula}\} \\ &\to \emptyset \vdash \phi \vee \neg\, \phi\end{aligned}$$

Formalizing propositional logic

There are three common structural rules:

- Weakening
- Exchange
- Contraction

Only weakening and exchange are needed.

A logic that rejects contraction is an affine logic.

Choice of logical connectives can affect dependencies on structural rules.

Biprovability relation

Definition

$\phi \sim \psi$ if and only if $\Gamma, \phi \vdash \psi$ and $\Gamma, \psi \vdash \phi$.

The relation is defined as a pair, so we define it as a product in Agda:

$$\_\sim\_ : \text{Formula} \to \text{Formula} \to \text{Type}$$
$$\phi \sim \psi = \Gamma :: \phi \vdash \psi \times \Gamma :: \psi \vdash \phi$$

This is an equivalence relation!

Lindenbaum-Tarski algebra

- The Lindenbaum-Tarski algebra for propositional logic is the quotient algebra obtained by quotienting the algebra of formulas by the equivalence relation $\sim$.
- The algebraic operations in the Lindenbaum-Tarski algebra are derived from the logical connectives present in the underlying logical system.
- These operations allow for the manipulation of formulas within the algebraic structure.

Formalizing Lindenbaum-Tarski algebra

Define Lindenbaum-Tarski algebra in Cubical Agda using the existing definition of set quotients.

$$\mathsf{LindenbaumTarski} : \mathsf{Type}$$
$$\mathsf{LindenbaumTarski} = \mathsf{Formula}\ /\ \_\!\sim\!\_$$

Formalizing Lindenbaum-Tarski algebra

Operations on the equivalence classes

$$\wedge/ \quad \vee/ \quad \neg/$$

To define these operations in Agda we made use of already existing definitions in the agda/cubical library.

Formalizing Lindenbaum-Tarski algebra

```
_∧/_ : LindenbaumTarski → LindenbaumTarski → LindenbaumTarski
A ∧/ B = setQuotBinOp ∼-refl ∼-refl _∧_ ∼-respects-∧ A B
```

If $\phi \sim \phi'$ and $\psi \sim \psi'$ then $\phi \wedge \psi \sim \phi' \wedge \psi'$.

```
∼-respects-∧ : ∀ (ϕ ϕ' ψ ψ' : Formula)
               → ϕ ∼ ϕ'
               → ψ ∼ ψ'
               → (ϕ ∧ ψ) ∼ (ϕ' ∧ ψ')
∼-respects-∧ ϕ ϕ' ψ ψ' (x₁ , x₂) (y₁ , y₂) =
              ∧-I (cut (∧-E₁ (axiom Z)) x₁) (cut (∧-E₂ (axiom Z)) y₁) ,
              ∧-I (cut (∧-E₁ (axiom Z)) x₂) (cut (∧-E₂ (axiom Z)) y₂)
```

Disjunction and negation are defined similarly.

The Lindenbaum-Tarski algebra is a Boolean algebra

#TODO

- A Boolean algebra is a complemented distributive lattice
- LT is complemented distributive lattice
- Formalizing this in Cubical Agda (+superweakening)

Applications

#TODO

- Soundness?
- Usefulness of doing algebra on logic?