Formal Construction of Set-Theoretic Models for an Extended Calculus of Constructions

joint work with Chad E. Brown

Jonas Kaiser

UdS, Graduate School of Computer Science

May 11, 2012
Outline

Overview

ECC

Tarski-Grothendieck Set Theory

Model Construction

Soundness

Goals
FORMALISED IN COQ
SET-THEORETIC MODEL
CONSISTENCY

TG
Proof Irrelevant
Classical
LOGIC
TYPE THEORY
(classical)

CHC
Coq, et al.

CIC / ECC
FORMALISED IN COQ
SET-THEORETIC MODEL
TG
Proof Irrelevant
Classical
LOGIC
TYPE THEORY (classical)
CHC Coq, et al.
CIC / ECC
CONSISTENCY
LOGIC

TYPE THEORY (classical)

CONSISTENCY

SET-THEORETIC MODEL

CHC

Coq, et al.

CIC / ECC
FORMALISED IN COQ LOGIC

TYPE THEORY (classical)

CHC

Coq, et al.

CIC / ECC

CONSISTENCY

Proof Irrelevant Classical

SET-THEORETIC MODEL

TG
FORMALISED IN COQ

LOGIC

TYPE THEORY (classical)

CHC

Coq, et al.

CIC / ECC

CONSISTENCY

SET-THEORETIC MODEL

TG

Proof Irrelevant Classical

FORMALISED IN COQ
Luo’s Extended Calculus of Constructions [4]

Term structure

- The kinds Prop and Type$_0$, Type$_1$, Type$_2$, … are terms
- Variables ($x, y, \ldots$) are terms
- Let $M, N, A$ and $B$ be terms, then

\[
\Pi x : A, B \mid \lambda x : A. N \mid MN |
\]

\[
\Sigma x : A, B \mid \text{pair}_{\Sigma x : A, B}(M, N) \mid \pi_1(M) \mid \pi_2(M)
\]

are terms
Luo’s Extended Calculus of Constructions [4]

Term structure

- The kinds Prop and Type$_0$, Type$_1$, Type$_2$, … are terms
- Variables ($x, y, \ldots$) are terms
- Let $M, N, A$ and $B$ be terms, then
  \[
  \Pi x : A, ~ B \mid \lambda x : A. ~ N \mid M N \mid \\
  \Sigma x : A, ~ B \mid \text{pair}_{\Sigma x : A, B} (M, N) \mid \pi_1 (M) \mid \pi_2 (M)
  \]
  are terms

Properties

- Strongly normalising
- No strong sums in Prop (would lead to inconsistency)
- Kinds are (fully) cumulative:
  \[
  \text{Prop} \preceq \text{Type}_0 \\
  \text{Type}_n \preceq \text{Type}_{n+1}
  \]
Tarski-Grothendieck set theory: ZFC & GU

\forall x, \ x \notin \emptyset

x \in \{a, b\} \iff x = a \lor x = b

x \in \bigcup A \iff \exists X \in A, \ x \in X

y \in \{F x \mid x \in X\} \iff \exists z, \ z \in X \land y = F z

X \in \mathcal{P}(A) \iff X \subseteq A

X = Y \iff X \subseteq Y \land Y \subseteq X

(\forall X, \ (\forall x \in X, \ P x) \rightarrow P X) \rightarrow \forall X, \ P X

Choice: should follow from \( \varepsilon \) in meta theory
Tarski-Grothendieck set theory: ZFC & GU

\[ \forall x, x \notin \emptyset \]
\[ x \in \{a, b\} \iff x = a \lor x = b \]
\[ x \in \bigcup A \iff \exists X \in A, x \in X \]
\[ y \in \{F x \mid x \in X\} \iff \exists z, z \in X \land y = Fz \]
\[ X \in \mathcal{P}(A) \iff X \subseteq A \]
\[ X = Y \iff X \subseteq Y \land Y \subseteq X \]
\[ (\forall X, (\forall x \in X, P x) \rightarrow P X) \rightarrow \forall X, P X \]

Choice: should follow from \( \varepsilon \) in meta theory

Grothendieck Universes

- Transitive set \((X \in U, x \in X \implies x \in U)\)
- Closed under above operators (e.g. \(X \in U \implies \mathcal{P}(X) \in U\))
- For every \(X\) there is a least universe \(U := G_X\) such that \(X \in G_X\)
- Implies infinity (\(G_\emptyset\) is inf.)
Model Construction

Abstract Model

- Some generic set constructions:
  * singletons, indexed unions, separation, ordered pairs

Concrete Model
Model Construction

Abstract Model

- Some generic set constructions: *singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:

  \[
  [[\text{Prop}]] := 2 \\
  [[\text{Type}_0]] := G_0
  \]

Concrete Model
Model Construction

Abstract Model

- Some generic set constructions: *singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:
  
  \[
  [[\text{Prop}]] := 2 \\
  [[\text{Type}_0]] := G_0 
  \]

- Inhabitance results to reflect validity of typing rules

Concrete Model
Model Construction

Abstract Model

- Some generic set constructions: 
  *singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:

\[
\begin{align*}
[[\text{Prop}]] & := 2 \\
[[\text{Type}_0]] & := G_\emptyset
\end{align*}
\]

- Inhabitance results to reflect validity of typing rules

Concrete Model

- Formalise syntax, environments, typing rules
Model Construction

Abstract Model

- Some generic set constructions: 
  * singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:

  \[
  [[\text{Prop}]] := 2 \\
  [[\text{Type}_0]] := G_0
  \]

- Inhabitance results to reflect validity of typing rules

Concrete Model

- Formalise syntax, environments, typing rules
- PTS-style or JE conversion formulation
Model Construction

Abstract Model

- Some generic set constructions: 
  * singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:
  \[
  [[\text{Prop}]] := 2 \\
  [[\text{Type}_0]] := G_0
  \]
- Inhabitance results to reflect validity of typing rules

Concrete Model

- Formalise syntax, environments, typing rules
- PTS-style or JE conversion formulation
- State soundness . . .
Model Construction

Abstract Model

- Some generic set constructions: *singletons, indexed unions, separation, ordered pairs*
- Specific set constructions to reflect parts of ECC:
  
  \[
  \begin{align*}
  \llbracket \text{Prop} \rrbracket & := 2 \\
  \llbracket \text{Type}_0 \rrbracket & := G_{\emptyset}
  \end{align*}
  \]

- Inhabitance results to reflect validity of typing rules

Concrete Model

- Formalise syntax, environments, typing rules
- PTS-style or JE conversion formulation
- State soundness . . .
- . . . and prove it?
Soundness

- If $\Gamma \vdash M : A$ is derivable, then it is valid in the model:

  $$\forall \gamma \in [[\Gamma]], \ [[M]]_{\gamma} \in [[A]]_{\gamma}$$
Soundness

- If $\Gamma \vdash M : A$ is derivable, then it is valid in the model:
  \[ \forall \gamma \in \llbracket \Gamma \rrbracket, \ [M]_\gamma \in [A]_\gamma \]

- Soundness follows, if all typing rules preserve validity.
Soundness

- If $\Gamma \vdash M : A$ is derivable, then it is valid in the model:

  $\forall \gamma \in \llbracket \Gamma \rrbracket, \llbracket M \rrbracket_\gamma \subseteq \llbracket A \rrbracket_\gamma$

- Soundness follows, if all typing rules preserve validity.

- Consider the Application rule:

  $\Gamma \vdash M : \Pi x : A, \ B \quad \Gamma \vdash N : A$

  $\Gamma \vdash M \ N : B \ [x := N]$
Soundness

- If $\Gamma \vdash M : A$ is derivable, then it is valid in the model:

  $$\forall \gamma \in [[\Gamma]], \ [[M]]_{\gamma} \in [[A]]_{\gamma}$$

- Soundness follows, if all typing rules preserve validity.

- Consider the Application rule:

  $$\begin{array}{c}
  \Gamma \vdash M : \Pi x : A, \ B \\
  \Gamma \vdash N : A
  \end{array}
  \quad
  \begin{array}{c}
  \Gamma \vdash M \ N : B [x := N]
  \end{array}$$

- We assume $\gamma \in [[\Gamma]], \ [[M]]_{\gamma} \in [[\Pi x : A, \ B]]_{\gamma}$ and $[[N]]_{\gamma} \in [[A]]_{\gamma}$
Soundness

- If $\Gamma \vdash M : A$ is derivable, then it is valid in the model:

$$\forall \gamma \in \llbracket \Gamma \rrbracket, \; \llbracket M \rrbracket_\gamma \in \llbracket A \rrbracket_\gamma$$

- Soundness follows, if all typing rules preserve validity.

- Consider the Application rule:

$$\frac{\Gamma \vdash M : \Pi x : A, \; B \quad \Gamma \vdash N : A}{\Gamma \vdash M \; N : B \; [x := N]}$$

- We assume $\gamma \in \llbracket \Gamma \rrbracket$, $\llbracket M \rrbracket_\gamma \in \llbracket \Pi x : A, \; B \rrbracket_\gamma$ and $\llbracket N \rrbracket_\gamma \in \llbracket A \rrbracket_\gamma$

- We have to show $\llbracket M \; N \rrbracket_\gamma \in \llbracket B \; [x := N] \rrbracket_\gamma$
Soundness, JE vs. PTS

\[
\begin{align*}
\Gamma \vdash_{\text{JE}} M : A \\
\Gamma \vdash_{\text{JE}} A = B : \text{Type}_i \\
\Gamma \vdash_{\text{JE}} M : B \\
\end{align*}
\]

(conv) \[
\begin{align*}
\Gamma \vdash_{\text{PTS}} M : A \\
\Gamma \vdash_{\text{PTS}} B : \text{Type}_i \\
\Gamma \vdash_{\text{PTS}} M : B \\
\end{align*}
\]

\( A \approx B \)

The JE case?

The PTS case?

Are they equivalent?
Soundness, JE vs. PTS

\[
\begin{align*}
\Gamma \vdash_{JE} M &: A \\
\Gamma \vdash_{JE} A = B &: \text{Type}_i \\
\hline
\Gamma \vdash_{JE} M &: B
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash_{PTS} M &: A \\
\Gamma \vdash_{PTS} B &: \text{Type}_i \\
\hline
\Gamma \vdash_{PTS} M &: B
\end{align*}
\]

The JE case?

Solved (e.g. Barras [2], Lee & Werner [3]).

The PTS case?

Are they equivalent?
Soundness, JE vs. PTS

\[
\frac{\Gamma \vdash_{JE} M : A}{\Gamma \vdash_{JE} A = B : \text{Type}_i} \quad \frac{\Gamma \vdash_{PTS} M : A}{\Gamma \vdash_{PTS} B : \text{Type}_i} \quad A \simeq B
\]

(conv)

The JE case?

Solved (e.g. Barras [2], Lee & Werner [3]).

The PTS case?

Unsolved, circularity problem (Miquel & Werner [5])

Are they equivalent?
Soundness, JE vs. PTS

\[
\frac{\Gamma \vdash_{JE} M : A}{\Gamma \vdash_{JE} A = B : Type_i} \quad \frac{\Gamma \vdash_{PTS} M : A}{\Gamma \vdash_{PTS} B : Type_i}
\]

The JE case?
Solved (e.g. Barras [2], Lee & Werner [3]).

The PTS case?
Unsolved, circularity problem (Miquel & Werner [5]).

Are they equivalent?
Unknown in the general case, approximations exist.
Results so Far . . .

- Adams [1]: equivalent, given uniqueness of types
Results so Far . . .

- Adams [1]: equivalent, given uniqueness of types
- Miquel & Werner [5]: circumvent problems in PTS case with syntactic annotations to ensure well-sortedness
Results so Far . . .

- Adams [1]: equivalent, given uniqueness of types
- Miquel & Werner [5]: circumvent problems in PTS case with syntactic annotations to ensure well-sortedness
- Pagano, Coquand et al. (02/2012): equivalent, when dropping impredicativity (norm. by eval.)
Model Properties of Interest

Consistency

- A statement is consistent when we can exhibit a satisfying model
- We construct a proof-irrelevant, classical model
- PI, DN, PE, FE, XM, ... should be satisfied in the model
Model Properties of Interest

Consistency
- A statement is consistent when we can exhibit a satisfying model
- We construct a proof-irrelevant, classical model
- PI, DN, PE, FE, XM, ... should be satisfied in the model

Independence
- A statement is independent when it is consistent but not provable
- To refute provability, exhibit a non-satisfying model
- E.g. existence of an infinite type in Type$_0$, formally

\[
\exists X : \text{Type}_0, \ \exists f : X \to X, (\exists x : X, \forall y : X, fy \neq x) \land \\
(\forall y z : X, fy = fz \rightarrow y = z)
\]
Mutual Consistency of Standard Library

Mutual Consistency

- (Γ, A consistent) ∧ (Γ, B consistent) ⇒ (Γ, A, B consistent)
- XM and ¬PI are both separately consistent with CiC . . .
- . . . but {XM, ¬PI} is inconsistent with CiC
Mutual Consistency of Standard Library

Mutual Consistency

- \((\Gamma, A \text{ consistent}) \land (\Gamma, B \text{ consistent}) \Rightarrow (\Gamma, A, B \text{ consistent})\)
- XM and \(\neg\text{PI}\) are both \textit{separately} consistent with CiC . . .
- . . . but \(\{XM, \neg\text{PI}\}\) is inconsistent with CiC

ECC representation of CiC axioms

- Consider \(P : \text{Prop}_{\text{CiC}}\), find suitable \(Q : \text{Prop}_{\text{ECC}}\)
- such that \(\vdash_{\text{CiC}} Q \leftrightarrow P\) (write \(Q \overset{\text{ECC}}{\iff} P\))
Mutual Consistency of Standard Library

Mutual Consistency

- \((\Gamma, A \text{ consistent}) \land (\Gamma, B \text{ consistent}) \Rightarrow (\Gamma, A, B \text{ consistent})\)
- XM and \(\neg\text{PI}\) are both separately consistent with CiC . . .
- . . . but \(\{\text{XM, } \neg\text{PI}\}\) is inconsistent with CiC

ECC representation of CiC axioms

- Consider \(P : \text{Prop}_{\text{CiC}}\), find suitable \(Q : \text{Prop}_{\text{ECC}}\)
- such that \(\vdash_{\text{CiC}} Q \leftrightarrow P\) (write \(Q \overset{\text{ECC}}{\leftrightarrow} P\))

Example: Proof Irrelevance

- In Coq/CiC: \(\forall (P:\text{Prop}) (p1\ p2:P), p1 = p2.\)
- In ECC: \(\Pi P : \text{Prop}, \Pi u : P, \Pi v : P, u =_P v\)
- Where \(u =_P v := \Pi R : P \rightarrow \text{Prop}, R u \rightarrow R v\)
- The abstract version of \(u =_P v\) an Coq’s = provably coincide.
Mutual Consistency of Standard Library

Axioms in the Library
Mutual Consistency of Standard Library

Axioms in the Library

**Classical**
- classical
- functional-extensionality
- Extensionality_Ensembles
Mutual Consistency of Standard Library

Axioms in the Library

Classical  classical, functional_extensionality, Extensionality_Ensembles

Choice  epsilon_statement, constructive_(in)definite_description, dependent_unique_choice, relational_choice
Mutual Consistency of Standard Library

Axioms in the Library

**Classical** classical, functional_extensionality, Extensionality_Esembles

**Choice** epsilon_statement, constructive_(in)definite_description, dependent_unique_choice, relational_choice

**PI** proof_irrelevance, eq_rect_eq, JMeq_eq
Mutual Consistency of Standard Library

Axioms in the Library

**Classical** classical, functional_extensionality, Extensionality_Ensembles

**Choice** epsilon_statement, constructive_(in)definite_description, dependent_unique_choice, relational_choice

**PI** proof_irrelevance, eq_rect_eq, JMeq_eq

**Reals** archimed, completeness, Rplus_assoc, . . .
Thesis Aims

For my thesis I want to . . .

- complete two abstract models
- formalise ECC concretely with JE conversion
- proof soundness for this scenario
- and show mutual consistency of the standard library

Given spare time . . .

- I’d like to investigate the PTS vs. JE problem
References

Robin Adams.
Pure type systems with judgemental equality.

Bruno Barras.
Sets in Coq, Coq in Sets.

Gyesik Lee and Benjamin Werner.
Proof-Irrelevant Model of CC with Predicative Induction and Judgmental Equality.

Zhaohui Luo.
ECC, an Extended Calculus of Constructions.

Alexandre Miquel and Benjamin Werner.
The Not So Simple Proof-Irrelevant Model of CC.
Appendix
Consistency

- Certain types are not inhabited
- $A$ provable ::= $\exists \mathcal{D}, \vdash \mathcal{D} : A$
- $\Gamma$ consistent ::= $\neg \exists \mathcal{D}, \Gamma \vdash \mathcal{D} : \bot$
- We take $\bot : \text{Prop} ::= \forall P : \text{Prop}, \ P$
- ECC is consistent (Proof via Strong Normalisation)
- XM, PE, FE, PI are consistent additions to ECC & CiC
- Set-theoretic Models: $\llbracket A \rrbracket \neq \emptyset$
Mutual Consistency & Independence

Independence

- A consistent \([\not \vdash A]\), not (A provable) \([\not \vdash A]\)
- To refute provability: provide a model where \([A] = \emptyset\)
- XM is independent from ECC
- (Type0 contains inf. types, like \(\mathbb{N}\)) is independent from ECC

Mutual Consistency

- \((\Gamma, A \text{ consistent}) \land (\Gamma, B \text{ consistent}) \Rightarrow (\Gamma, A, B \text{ consistent})\)
- XM and \(\neg\Pi\) are both separately consistent with CiC . . .
- . . .but \(\{XM, \neg\Pi\}\) is inconsistent with CiC
Constructions in TG & Meta Theory

- Singleton Sets: \{x\}
- 1: \{\emptyset\} = \mathcal{P}(\emptyset)
- 2: \{\emptyset, 1\} = \mathcal{P}(1)
- Indexed Union: \bigcup_{i \in I} X_i
- Separation: \{x \in X \mid Px\}
- Ordered Pairs (Kuratowski): (a, b)_k
- Cartesian Product: A \times B

Related Lemmas
Introduction and elimination rules, correctness statements and useful equalities with respect to the special sets 0, 1 and 2.

Meta Theory
We use classical CiC with extensionality principles and Hilbert’s \(\varepsilon\).
The ECC Model

Kinds:

\[
[\text{Prop}] := 2
\]
\[
[\text{Type}_0] := \mathbb{G}_\emptyset
\]

Functions (using Aczel’s encoding):

\[
\text{ap } f \ x := \{y \mid (x, y) \in f\}
\]
\[
\text{lam } d \ F := \{(x, y) \mid x \in d \land y \in F \ x\}
\]
\[
\text{Pi } d \ Y := \{\text{lam } d \ F \mid \forall x \in d, F \ x \in Y \ x\}
\]

Strong Sums & Pairs:

- \[
\text{Sig } d \ Y := \text{lam } d \ Y
\]
- Pairs: \((a, b) := \{\{a\}, \{a, b\}\}\)
Preliminary Results

True and False

- Both are in \([\text{Prop}]\)
- \([\text{FALSE}]\) = 0
- \([\text{TRUE}]\) = 1

Leibniz Equality

- defined in object logic, one for each type level
- coincides with Coq’s equality on our meta type \(\text{set}\)
- asserts that domains are ok

Proof Irrelevance

- We have shown \([\text{PI}]\) = 1
- i.e. inhabited . . .
- . . . and in \([\text{Prop}]\)
Barras [2]

- Defines signatures for CC and CC\(_\omega\) models
- Works in an intuitionistic setting
- Models are proof-irrelevant
- Implements his signatures using HF and IZF
- Proves soundness of his signatures when using JE
- Obtains soundness for CC with Conversion via Adams
- Won’t work for CC\(_\omega\) since we lack *uniqueness of types*
- Fully formalised in Coq
- Models for our signatures also satisfy his signatures.
Werner, Lee & Miquel [5, 3]

- Initially compared proof theoretic strength of CiC and ZFC
- Models are proof-irrelevant
- Later mostly focused on the soundness problem.
- ‘Solved’ by syntactically annotating variables with sorts and dropping Prop ≤ Type₀
- Partially formalised in Coq
- They aim for CiC but exclude Inductive Propositions
Remarks on the Meta theory

- We work in Coq: CiC
- Add ClassicalFacts: relates XM, PD, PE, PI, …
- Add ClassicalProp: XM, DN, Peirce, PI, …
- Add FunctionalExtensionality: FE, …
- Add Epsilon: Hilbert’s $\varepsilon$, Church’s $\iota$
- CDP: $\forall P : \text{Prop}, \ P + (\neg P)$ follows from DN and $\varepsilon$.
- $\forall P : \text{Type}, \ P + (P \rightarrow \bot)$ follows from CDP and $\varepsilon$. 
Barras’ Framework

CC_Model (Barras)

ECC_Model (*) (Barras)

TG_CC_Model

TG_ECC_Model

TG_CC_Model_Spec

TG_ECC_Model_Spec

TG

TG_CC_Model

TG_ECC_Model

TG_CC_Model1

TG_ECC_Model1,2
\[ \exists X : \text{Type}_0, \ \exists f : X \to X, (\exists x : X, \ \forall y : X, fy \not= x) \land \\
(\forall y z : X, fy = fz \rightarrow y = z) \]

\(f\) is a function on \(X\) which is \textit{injective} but \textit{not surjective}. This implies that \(X\) is infinite.

- Not satisfied in the \([\text{Type}_0] := G_\emptyset\) model; (any injective \(f\) on \(X\) is also surjective – classical)
- Satisfied in the \([\text{Type}_0] := G_{G_\emptyset}\) model; Use \(X := G_\emptyset\) and \(f := P\)
What’s wrong with the standard graph encoding of functions?

- The function space \( T \to T \) contains exactly one element, the function mapping \( \emptyset \) to \( \emptyset \).
- with standard graph-encoding: \( [[T \to T]] = \{((\emptyset, \emptyset))\} \not\in 2 \)
- however, we want \( [[T \to T]] = 1 \in 2 \)
- but \( \emptyset \neq ((\emptyset, \emptyset)) \)!
- with the alternative function encoding, the two sides match up.