# Formal Proof of the Relative Consistency of ZF+$\neg$AC with Isabelle/ZF

Daiki Funane

Graduate School of Information Sciences, Tohoku University

(Draft of Master's Thesis)

December 25, 2024

# Abstract

This paper presents a formalization of the relative consistency of the negation of the Axiom of Choice (¬AC) with respect to Zermelo-Fraenkel set theory (ZF) using the proof assistant Isabelle/ZF. We use the countable transitive model (c.t.m.) approach, assuming the existence of a c.t.m. of ZF, and apply forcing to construct a symmetric extension called the basic Cohen model, which serves as a model of ZF + ¬AC. This work builds on previous formalizations, such as Gunther et al.'s work on forcing, and provides the first formalization of the relative consistency of ¬AC with ZF.

The proof is based on Karagila's lecture note and Jech's textbook, which align well with the c.t.m. approach. However, some parts required adopting alternative proof methods. The formalization process revealed insights into the challenges of formalizing set theory and the potential for future improvements.

# Acknowledgements

Oleg Kiselyov

# Contents

# Chapter 1

# Introduction

**Proof Assistants** Proof assistants are software tools designed to formally verify mathematical proofs, programs, and more. These systems are widely used in various fields, including mathematics [7, 10], cryptography [1, 6], hardware verification [5, 2], and program verification [26]. Operating on the basis of formal logic, they enable users to interactively construct proofs, which are then checked by the system. Well-known examples of proof assistants include Isabelle [29], Coq [30], Lean [4] Agda [27], and HOL Light [28], each offering unique features and strengths [19].

One of the key advantages of proof assistants is their ability to guarantee the correctness of proofs. In manually constructed proofs, errors can easily occur due to human mistakes, and even when the proof is correct, it can be difficult to verify its correctness. Proof assistants provide a solution to these problems by rigorously checking each logical step of given proof, ensuring its correctness.

In particular, these systems are effective in verifying complex proofs. For example, the Four Color Theorem [7] and Kepler's Conjecture [10] are well-known cases of theorems formally proven using proof assistants. These large-scale proofs were difficult for humans to verify manually, but their correctness was assured through formal verification. Similarly, in systems where safety is critical, formal verification of programs can significantly enhance their reliability. Notable examples of formally verified systems include the CompCert C compiler [18] and the seL4 microkernel [15].

Proof assistants are not limited to applications requiring complex proofs; they are also widely used in efforts to formalize various mathematical theorems. Fundamental mathematical results, such as those in group and ring theory or calculus, have been formalized by many proof assistants, and the scope of such formalizations continues to expand.

The results presented in this paper involve the formalization of theorems in axiomatic set theory using Isabelle. This also serves as an example of a formal proof of an argument using forcing.

**Axiomatic Set Theory** Set theory plays a crucial role as the foundation of modern mathematics. From natural numbers and functions to more advanced and complex concepts, mathematical ideas and theorems can basically be expressed within the framework of set theory. As a result, set theory functions as a common language in modern mathematics.

In the 19th century, set theory was devised by Cantor, bringing significant progress to mathematics. However, during its development, issues such as Russell's paradox emerged. Axiomatic set theory was developed to address these issues and to investigate which operations on sets are permissible. In axiomatic set theory, allowable operations are strictly defined by axioms, and the theory is constructed based on these axioms.

Over time, various axiom systems have been proposed, and the relationships and independence among these systems have been studied. Among the axiom systems in axiomatic set theory, the Zermelo-Fraenkel system (ZF) and the Neumann-Bernays-Gödel system (NBG) are well-known. This paper primarily focuses on the ZF and ZFC axiom systems.

The Zermelo-Fraenkel (ZF) axiom system is an extension of the axiom system proposed by Zermelo, further developed by Fraenkel, Skolem, von Neumann, and others. When the Axiom of Choice is included, the system is referred to as ZFC and is widely used. These axiom systems are defined within the framework of first-order logic and include infinitely many axioms specified by axiom schemas. It is well known that most mathematical concepts can be expressed within ZFC, and its properties have been widely studied.

Among these, the independence of the Axiom of Choice (AC) in ZF and the independence of the Continuum Hypothesis (CH) in ZFC are significant early results. These were proven by Cohen using a technique known as forcing.

While this paper provides a brief explanation of axiomatic set theory, a detailed exposition can be found in standard textbooks such as [16, 13].

**Relative Consistency and Independence** One of the interesting and important aspects of axiomatic set theory is that multiple "levels" of set theory appear within arguments. For example, by formalizing first-order logical formulas and formal proof systems such as sequent calculus within ZFC set theory, it is possible to build ZFC set theory internally within ZFC itself.

We also use ZF or ZFC externally and consider other set theories within them. In this paper, we often refer to the external system as the meta-level set theory and the one constructed within it as the object-level set theory.

Basically, when we say "$\varphi$ is provable in the axiom system $T$", $T$ refers to the object-level set theory, not the meta-level. In other words, it means that within the formal proof system constructed on the meta-level set theory, the axiom system $T$ and the formula $\varphi$, also constructed within the same system, can be proven.

A proposition $\varphi$ is said to be independent of an axiom system $T$ if neither $\varphi$ nor $\neg\varphi$ is provable in $T$. This is equivalent to saying that both the system $T + \varphi$ and $T + \neg\varphi$ are consistent, where $T + \varphi$ denotes the system obtained by adding $\varphi$ to $T$. Furthermore, in first-order logic, an axiom system is considered consistent if it has a model. Therefore, to prove that $\varphi$ is independent of $T$, it is sufficient to show that both $T + \varphi$ and $T + \neg\varphi$ have models.

However, when considering $T$ as systems like ZF or ZFC, it is impossible, within the current context, to demonstrate their consistency without any assumptions, due to

Gödel's incompleteness theorem. Instead, the consistency of the system of interest is proven by assuming the consistency of another axiom system. This property is referred to as relative consistency.

From the above, to be precise, the independence of AC in ZF is proven by assuming the consistency of ZF and showing that both ZF + AC and ZF + ¬AC have models. (However, in the current setting, the consistency of ZF and the consistency of ZFC are equivalent, so it is also valid to assume the consistency of ZFC.) Similarly, the independence of CH in ZFC is proven by assuming the consistency of ZFC and showing that both ZFC + CH and ZFC + ¬CH have models. Forcing is a powerful technique for carrying out such proofs.

**Forcing** Forcing is a technique first introduced by Cohen [3] to prove the independence of AC in ZF and the independence of CH in ZFC. It is a powerful method for exploring models of set theory, and it has been further refined by other researchers over time, leading to many significant results being established through its use. Forcing continues to play a crucial role in contemporary research in axiomatic set theory.

As will be explained in more detail in Chapter 2, forcing is a technique for extending models of set theory. Given a model $M$ of set theory and a poset $P$ within $M$, applying the forcing method produces the smallest extension $M[G]$ (called a generic extension) of $M$ that includes a set $G$ known as a generic filter on $P$. By carefully choosing $P$, it is possible to control certain properties that hold in $M[G]$. Furthermore, what holds or does not hold in $M[G]$ can be identified using the forcing relation.

For example, independence of CH is proven by assuming the consistency of ZFC to obtain a model of ZFC, and then extending it using forcing to construct a model of ZFC in which CH holds and another model in which ¬CH hold. In the case of AC, a submodel of a generic extension is constructed to obtain a model of ZF where AC does not hold.

**Previous Research on the Formalization of Axiomatic Set Theory** As with other areas of mathematics, the formalization of axiomatic set theory using proof assistants has also been advancing.

Paulson et al. formalized an extensive part of ZF set theory, including cardinal arithmetic [25], relativization [24], the reflection theorem [23], features for handling inductive definitions [20], and the relative consistency of AC with ZF [24] The proof of the relative consistency was achieved by constructing Gödel's constructible universe.

Building on these results, Gunther et al. formalized forcing and a proof of the independence of CH [8, 9] in Isabelle/ZF. In these formalizations, they used the countable transitive model (c.t.m.) approach, which we also adopt in this paper.

In Lean, Han and van Doorn also formalized forcing and the independence of CH [11] in Lean 3. using the Boolean-valued model approach, which is another approach to forcing. Additionally, in Lean 4, Holmes and Wilshaw formalized the complex parts of the consistency proof of Quine's New Foundations [12], ensuring the correctness of the proof.

Although these results have been formalized, the relative consistency of ¬AC with ZF has not yet been formalized.

**Our Contribution and Approach** In this paper, we formalize the proof of the relative consistency of ¬AC with ZF using Isabelle/ZF.

This work contributes to the formalization of axiomatic set theory and serves as a new example of the formalization using forcing, which is a crucial tool in set theory. It also highlights parts of the process that require significant effort and provides insights into how future formalizations could be made more efficient.

The primary reason we chose Isabelle/ZF for this formalization is that Isabelle/ZF is a mature proof assistant for ZF set theory, in particular, the formalization by [8] is a major advantage for this study. Although there is also a formalization of forcing in Lean 3 by Han and van Doorn [11], development for Lean 3 has already ended.

We adopted the c.t.m. approach for our proof to use Gunther et al.'s formalization, as in their independence proof of CH. Specifically, we show th relative consistency of ¬AC with ZF by assuming the existence of a c.t.m. of ZF and constructing a model of ZF+¬AC by forcing. This model is known as the basic Cohen model, which is a symmetric extension of assumed c.t.m. Constructing the basic Cohen model by forcing is one of traditional methods for proving the relative consistency of ¬AC with ZF.

Our proof is based on Karagila's lecture note [17] and Jech's book [14], as these resources align well with the c.t.m. approach. However, some parts of the proof required modification to fit the formalization in Isabelle/ZF.

The c.t.m. approach is a well-established method for relative consistency proofs in axiomatic set theory. This approach will be explained in detail in Section 2.1.4. The discussion regarding the validity of the c.t.m. approach, which is often omitted or only briefly mentioned in paper-based proofs, has not yet been formalized, and its formalization is desirable. Further details about this will be provided in Chapter 4.

**Synopsis** The remainder of this thesis is outlined as follows. In Chapter 2, we first explain set-theoretic concepts in Section 2.1. Section 2.2 outlines an informal proof (i.e., a proof on paper) of the relative consistency of ¬AC with ZF, which is formalized in Chapter 3. In Section 2.3, we provide an overview of Isabelle/ZF and highlight key results from prior work that are relevant to this study. Chapter 3 presents the formalization of the relative consistency proof of ZF+¬AC, building on the contents of Chapter 2. Finally, Chapter 4 concludes the thesis.

**Repository** Our source code is available at: `https://github.com/tarakojo/ZF_notAC`

# Chapter 2

# Preliminaries

## 2.1 Set-Theoretic Preliminaries

In this section, we introduce the concepts of set theory used in the formalization of this study. We discuss in ZFC, within which we explore ZF itself. Basically, our definition and theorems combine [16], [17], and [14] adapted to the formalized form in Isabelle/ZF.

### 2.1.1 ZF Set Theory and the Axiom of Choice

We introduce the axioms of ZF set theory and the Axiom of Choice (AC). We use first-order logic with the language of set theory, which consists of only two relation symbols $\in$ and $=$. Formulas involving other mathmatical operators that may appear are considered abbreviations for formulas in this language. Parentheses in formulas are omitted where no confusion arises.

**Definition 2.1.1.** *The axioms of ZF are the following statements:*

- *Extensionality:* $\forall x \forall y (\forall z (z \in x \leftrightarrow z \in y) \rightarrow x = y)$
- *Pairing:* $\forall x \forall y \exists z \forall w (w \in z \leftrightarrow w = x \vee w = y)$
- *Union:* $\forall x \exists y \forall z (z \in y \leftrightarrow \exists w (z \in w \wedge w \in x))$
- *Power set:* $\forall x \exists y \forall z (z \in y \leftrightarrow z \subseteq x)$
- *Infinity:* $\exists x (\emptyset \in x \wedge \forall y (y \in x \rightarrow y \cup \{y\} \in x))$
- *Regularity:* $\forall x (x \neq \emptyset \rightarrow \exists y (y \in x \wedge y \cap x = \emptyset))$
- *Infinity:* $\forall x (x \neq \emptyset \rightarrow \exists y (y \in x \wedge \forall z (z \in x \rightarrow z \notin y)))$
- *Separation:* $\forall p \forall x \exists y \forall z (z \in y \leftrightarrow z \in x \wedge \phi(z, p))$
- *Replacement:* $\forall p (\forall x \forall y \forall z (\phi(x, y, p) \wedge \phi(x, z, p) \rightarrow y = z) \rightarrow$ $\forall X \exists Y \forall y (y \in Y \leftrightarrow \exists x (x \in X \wedge \phi(x, y, p))))$

Where separation and replacement are axiom schemas, representing infinitely many axioms for each formula $\phi$ with an appropriate arity.

**Definition 2.1.2.** The Axiom of Choice *is the following statement:*
$\forall x \exists f(\text{"}f$ *is a function on* $x\text{"} \wedge \forall y(y \in x \rightarrow f(y) \in y))$

Where the phrase "f is a function on x" is also considered an abbreviation in the language of set theory. Additionally, we introduce the well-ordering theorem, as we treat AC in this form.

**Definition 2.1.3.** *We say that a linear ordering* $<$ *on a set* $P$ *is a* well-ordering *if, every non-empty subset of* $P$*, it has a least element.*

**Lemma 2.1.4.** *The Axiom of Choice is equivalent to the well-ordering theorem, which states that every set can be well-ordered.*

### 2.1.2 Forcing

Forcing is a tequnique used in proving relative consistency and independence. We introduce basic concepts of forcing in the context of the c.t.m. approach. In this approach, the relative consistency proof is achieved by using forcing to construct a extended model by adding new sets to an assumed c.t.m. Let $M$ be a c.t.m. of ZF and $(\mathbb{P}, \leq_{\mathbb{P}})$ be a notion of forcing, which is a pre-ordered set in $M$ with a maximum element $1_{\mathbb{P}}$.

**Definition 2.1.5.** *We define* $M^{\mathbb{P}}$*, the set of* $\mathbb{P}$-names*, by transfinite recursion on ordinals:*

1. $M_0^{\mathbb{P}} = \emptyset$
2. $M_{\alpha+1}^{\mathbb{P}} = \mathcal{P}^M(M_\alpha^{\mathbb{P}} \times \mathbb{P})$
3. $M_\alpha^{\mathbb{P}} = \bigcup_{\beta<\alpha} M_\beta^{\mathbb{P}}$ *for a limit ordinal* $\alpha$
4. $M^{\mathbb{P}} = \bigcup_{\alpha \in \mathrm{Ord}} M_\alpha^{\mathbb{P}}$

Where $\mathcal{P}^M$ denotes the power set operation in $M$. We often write a $\mathbb{P}$-name with a dot, e.g., $\dot{x}$. The least $\alpha$ such that $\dot{x} \in M_{\alpha+1}^{\mathbb{P}}$ is called the $\mathbb{P}$-*rank* of $\dot{x}$. This allows us to define functions and relations by recursion on $\mathbb{P}$-names.

**Definition 2.1.6.**

1. *We say that* $D \subseteq \mathbb{P}$ *is* dense *if, for every* $p \in \mathbb{P}$*, there exists* $q \in D$ *such that* $q \leq_{\mathbb{P}} p$.
2. *We say that* $G \subseteq \mathbb{P}$ *is a* filter *if following conditions hold:*
   - *If* $p \in G$*,* $q \in \mathbb{P}$*, and* $p \leq_{\mathbb{P}} q$*, then* $q \in G$
   - *If* $p, q \in G$*, there exists* $r \in G$ *such that* $r \leq_{\mathbb{P}} p$ *and* $r \leq_{\mathbb{P}} q$
3. *We say that* $G \subseteq$ *is* generic filter *on* $\mathbb{P}$ *if* $G$ *is a filter and for any dense* $D \subseteq \mathbb{P}$*,* $D \cap G \neq \emptyset$.

The following lemma shows that a generic filter actually exists.

**Lemma 2.1.7.** *For any $p \in \mathbb{P}$, there exists a generic filter $G$ on $\mathbb{P}$ such that $p \in G$.*

**Definition 2.1.8.** *Let $G$ be a generic filter on $\mathbb{P}$ and $\dot{x} \in M^{\mathbb{P}}$. We define the* interpretation *of $\dot{x}$ denoted by $\dot{x}^G$ recursively with respect to the $\mathbb{P}$-rank of $\dot{x}$:*

$$\dot{x}^G = \{\dot{y}^G \mid \exists p \in G(\langle \dot{y}, p \rangle \in \dot{x})\}$$

We call a $\mathbb{P}$-name whose interpretation is a set $x$ a name of $x$ and denote it by $\dot{x}$. Note that a single set may have multiple names.

**Definition 2.1.9.** *Let $G$ be a generic filter on $\mathbb{P}$. We define a* generic extension $M[G]$ *as $\{x^G \mid \dot{x} \in M^{\mathbb{P}}\}$.*

**Theorem 2.1.10.** *Let $G$ be a generic filter on $\mathbb{P}$. Then, $M[G]$ is the smallest c.t.m. of ZF extending $M$ and containing $G$.*

By choosing $\mathbb{P}$ appropriately, we can construct $M[G]$ with various properties. What holds or does not hold in $M[G]$ can be identified using the forcing relation. The forcing relation is defined recursively on formulas in a forcing language. The forcing language is an extension of the language of set theory by adding the elements of $M^{\mathbb{P}}$ as constants.

**Definition 2.1.11.** *We define the* forcing relation $\Vdash$ *for formulas in the forcing language and $p \in \mathbb{P}$ inductively* [1] *as follows:*

1. $p \Vdash \dot{x} = \dot{y} \Leftrightarrow \forall \dot{z} \in \mathrm{dom}(\dot{x}) \cup \mathrm{dom}(\dot{y}) \, \forall q \leq_{\mathbb{P}} p \, (q \Vdash \dot{z} \in \dot{x} \leftrightarrow q \Vdash \dot{z} \in \dot{y})$
2. $p \Vdash \dot{x} \in \dot{y} \Leftrightarrow \forall q \leq_{\mathbb{P}} p \, \exists r \leq_{\mathbb{P}} q \, \exists s \in \mathbb{P} \, \exists \dot{z} \in M^{\mathbb{P}} \, (\langle \dot{z}, s \rangle \in \dot{y} \wedge r \leq_{\mathbb{P}} s \wedge r \Vdash \dot{x} = \dot{z})$
3. $p \Vdash \phi \wedge \psi \Leftrightarrow p \Vdash \phi \wedge p \Vdash \psi$
4. $p \Vdash \neg\phi \Leftrightarrow \forall q \leq_{\mathbb{P}} p \, \neg(q \Vdash \phi)$
5. $p \Vdash \exists x \phi(x) \Leftrightarrow \forall q \leq_{\mathbb{P}} p \, \exists r \leq_{\mathbb{P}} q \, \exists \dot{x} \in M^{\mathbb{P}} \, (r \Vdash \phi(\dot{x}))$

We say that *$p$ forces $\phi$* if $p \Vdash \phi$.

**Theorem 2.1.12** (The Truth Lemma)**.** *Let $G$ be a generic filter on $\mathbb{P}$, $\varphi$ be a formula, and $\dot{x} \in M^{\mathbb{P}}$, then*

$$M[G] \vDash \varphi(\dot{x}^G) \Leftrightarrow \exists p \in G \, (p \Vdash \varphi(\dot{x}))$$

**Corollary 2.1.13.** *Let $p \in \mathbb{P}$, $\varphi$ be a formula and $\dot{x} \in M^{\mathbb{P}}$, then*

$$p \Vdash \varphi(\dot{x}) \Leftrightarrow \textit{for any generic filter } G \textit{ containing } p, M[G] \vDash \varphi(\dot{x}^G)$$

[1] Specifically, the forcing relation is first defined for atomic formulas $\dot{x} = \dot{y}$ and $\dot{x} \in \dot{y}$ by mutual recursion, inductively on the $\mathbb{P}$-rank of $\dot{x}$ and $\dot{y}$. Then, the definition is extended to all formulas in the forcing language by induction on the complexity of formulas.

### 2.1.3 Symmetric Extensions

Symmetric extensions are substructures of generic extensions of a given c.t.m. of ZF and are formed by interpreting only the hereditarily symmetric names. Let $M$ be a c.t.m. of ZF, $(\mathbb{P}, \leq_{\mathbb{P}})$ be a pre-ordered set in $M$ with the maximum element $1_{\mathbb{P}}$.

**Definition 2.1.14.** *We say that $\pi : \mathbb{P} \to \mathbb{P}$ is an* automorphism *if for all $p, q \in \mathbb{P}$, $p \leq_{\mathbb{P}} q \Leftrightarrow \pi p \leq_{\mathbb{P}} \pi q$. $\pi$ induces an bijection on $\mathbb{P}$-names defined recursively as follows:*

$$\pi\dot{x} = \{\langle \pi\dot{y}, \pi p\rangle \mid \langle \dot{y}, p\rangle \in \dot{x}\}$$

**Definition 2.1.15.** *Let $\mathcal{G}$ be a group of automorphisms of $\mathbb{P}$. We say that $\mathcal{F}$ is a* normal filter *on $\mathcal{G}$ if the following conditions hold:*

1. *$\mathcal{F}$ is non-empty family of subgroups of $\mathcal{G}$.*
2. *$\mathcal{F}$ is closed under finite intersections and supergroups.*
3. *For every $H \in \mathcal{F}$ and $\pi \in \mathcal{G}$, $\pi H \pi^{-1} \in \mathcal{F}$.*

We fix a group of automorphisms $\mathcal{G}$ of $\mathbb{P}$ and a normal filter $\mathcal{F}$ on $\mathcal{G}$.

**Definition 2.1.16.** *For every $\mathbb{P}$-name $\dot{x}$, let* $\mathrm{sym}(\dot{x}) = \{\pi \in \mathcal{G} \mid \pi\dot{x} = \dot{x}\}$. *We say that $\mathbb{P}$-name $\dot{x}$ is* symmetric *if* $\mathrm{sym}(\dot{x}) \in \mathcal{F}$.

**Definition 2.1.17.** *We define the set of all hereditarily symmetric names* HS *recursively as follows:*

1. $0 \in \mathrm{HS}$
2. *if* $\mathrm{dom}(\dot{x}) \subseteq \mathrm{HS}$ *and $\dot{x}$ is symmetric, then* $\dot{x} \in \mathrm{HS}$

**Definition 2.1.18.** *Let $G$ be a generic filter on $\mathbb{P}$. The set* $\mathrm{HS}^G = \{\dot{x}^G \mid \dot{x} \in \mathrm{HS}\}$ *is called a* symmetric extension *of $M$.*

**Theorem 2.1.19.** *Let $G$ be a generic filter on $\mathbb{P}$. Then, the symmetric extension* $\mathrm{HS}^G$ *is a c.t.m. of ZF and a substructure of $M[G]$.*

**Definition 2.1.20.** *The relativized forcing relation $\Vdash_{\mathrm{HS}}$ is defined as the forcing relation $\Vdash$ with $M^{\mathbb{P}}$ in its definition replaced by* HS.

The relation $\Vdash_{\mathrm{HS}}$ acts as the forcing relation for symmetric extensions.

**Theorem 2.1.21.** *Let $G$ be a generic filter on $\mathbb{P}$, $\mathcal{N}$ be a symmetric extension generated by $G$, $\varphi$ be a formula, and $\dot{x} \in \mathrm{HS}$, then*

$$\mathcal{N} \vDash \varphi(\dot{x}) \Leftrightarrow \exists p \in G\ (p \Vdash_{\mathrm{HS}} \varphi(\dot{x}))$$

**Corollary 2.1.22.** *Let $p \in \mathbb{P}$, $\varphi$ be a formula, and $\dot{x} \in \mathrm{HS}$, then*

$$p \Vdash_{\mathrm{HS}} \varphi(\dot{x}) \Leftrightarrow \textit{for any generic filter } G \textit{ containing } p, \mathcal{N} \vDash \varphi(\dot{x}^G)$$

*Where $\mathcal{N}$ is the symmetric extension generated by $G$.*

The following lemmas also holds for the forcing relation $\Vdash$, but we only state them for $\Vdash_{\mathrm{HS}}$.

**Lemma 2.1.23.** *Let $p, q \in \mathbb{P}$, $\varphi$ be a formula. If $q \leq_{\mathbb{P}} p$ and $p \Vdash_{\mathrm{HS}} \varphi$, then $q \Vdash_{\mathrm{HS}} \varphi$.*

**Lemma 2.1.24** (The Symmetry Lemma)**.** *Let $\pi$ be an automorphism of $\mathbb{P}$, $\dot{x}_0, \ldots, \dot{x}_n \in$ HS, and $\varphi$ be a formula, then*

$$p \Vdash_{\mathrm{HS}} \varphi(\dot{x}_0, \ldots, \dot{x}_n) \Leftrightarrow \pi p \Vdash_{\mathrm{HS}} \varphi(\pi\dot{x}_0, \ldots, \pi\dot{x}_n)$$

### 2.1.4 The c.t.m. Approach

This section explains the c.t.m. approach. This approach is a well-established method for proving relative consistency in axiomatic set theory, as it is covered in [16].

Specifically, the approach assumes the existence of a countable transitive model (c.t.m.) of an axiom system $S$ of set theory and show the existence of a c.t.m. for another axiom system $T$, thereby proving the relative consistency of $T$ with respect to $S$. This approach is useful because it allows discussions to proceed without worrying about the complicated details that can arise.

It should be noted that discussions are needed to confirm the validity of this approach. By definition, the relative consistency of $T$ with respect to $S$ means that if $S$ is consistent, then $T$ is also consistent. Assuming that $S$ is consistent, by the completeness theorem, we can prove that $S$ has a model. However, the existence of a c.t.m. of $S$ cannot be necessarily derived from the consistency of $S$.

Nonetheless, the c.t.m. approach is valid for proving the relative consistency. The following presents a discussion on the validity of the approach. In accordance with the content of this paper, we proceed with the discussion by letting $S$ be ZF and $T$ be ZF + $\neg$AC. However, the discussion below is also applicable to other cases.

First, we introduce some concepts, following Jech [13], chapter 6 and 12.

**Proposition 2.1.25** (Reflection Principle)**.** *Let $\varphi(x_1, \ldots, x_n)$ be a formula. There is a transitive set $M$ such that*

$$\varphi^M(x_1, \ldots, x_n) \leftrightarrow \varphi(x_1, \ldots, x_n)$$

*for every $x_1, \ldots, x_n \in M$.*

Where $\varphi^M$ denotes the relativization of $\varphi$ to $M$. In other words, $\varphi^M$ is the formula obtained by replacing every quantifier $\forall x$ by $\forall x \in M$ and $\exists x$ by $\exists x \in M$. Note that this is a theorem schema for every formula $\varphi$.

**Definition 2.1.26.** *Let $E$ be a binary relation on a set $P$.*

- *We say thet $E$ is* well-founded *if every non-empty subset of $P$ has an $E$-minimal element.*

- *We say that $E$ is extensional if:*

  $$\mathrm{ext}_E(x) \neq \mathrm{ext}_E(y)$$

  *whenever $x$ and $y$ are distinct elements of $P$, where*

  $$\mathrm{ext}_E(x) = \{z \in P \mid z \; E \; x\}$$

**Theorem 2.1.27** (Mostowski's Collapsing Theorem)**.** *If $E$ is a well-founded and extensional relation on a set $P$, then there is a transitive set $M$ and an isomorphism $\pi$ between $(P, E)$ and $(M, \in)$.*

**Theorem 2.1.28** (Löwenheim-Skolem Theorem)**.** *Every infinite model for a countable language has a countable elementary submodel.*

Using these, we can prove the following proposition:

**Proposition 2.1.29.** *Let $\Gamma$ be a finite fragment of ZF axioms, including the Axiom of Extensionality, then there exists a c.t.m. of $\Gamma$.*

*Proof.* Let $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$ and $\psi = \varphi_1 \wedge \ldots \wedge \varphi_n$. By reflection principle, there exists a set $M$ such that $\psi^M \leftrightarrow \psi$. Then, $\mathcal{M} = (M, \in|_{M\times M})$ satisfies $\psi$. In other words, $\mathcal{M}$ is a model of $\Gamma$. By Theorem 2.1.28, there exists a countable elementary submodel $\mathcal{N} = (N, \in|_{N\times N})$ of $\mathcal{M}$. $\mathcal{N}$ satisfies the Axiom of Extensionality since $\Gamma$ includes it. Additionary, by the Axiom of Regularity, every non-empty subset of $N$ has an $\in$-minimal element. Therefore, $\in|_{N\times N}$ is well-founded and extensional. By Mostowski's Collapsing Theorem, there exists a transitive set $N_0$ and an isomorphism between $(N, \in|_{N\times N})$ and $(N_0, \in)$. Thus, $N_0$ is a c.t.m. of $\Gamma$. □

Here, we use this to explain the validity of the c.t.m. approach. First, let us assume the existence of a c.t.m. of ZF and suppose that we can prove the existence of a c.t.m. of ZF+¬AC by forcing. It should also be possible to prove the following by modifying the proof.

**Proposition 2.1.30.** *Let $\Gamma$ be a finite fragment of ZF, there exists a finite fragment $\Delta$ of ZF+¬AC such that if a c.t.m. of $\Gamma$ exists, then a c.t.m. of $\Delta$ also exists.*

This is because, in the original proof, the axiom of ZF used to formalize forcing are finite, and the proof that a c.t.m. of ZF+¬AC satisfies each axiom of ZF+¬AC should only require a finite number of axioms of ZF.

By combining Propositions 2.1.29 and 2.1.30, we can prove the following[2].

**Proposition 2.1.31.** *Let $\Delta$ be a finite fragment of ZF+¬AC, there exists a c.t.m. of $\Delta$. In other words, $\Delta$ is consistent.*

Here, let us prove the relative consistency of ZF+¬AC by contradiction. Assume that ZF+¬AC is inconsistent. Then, since ZF+¬AC proves $\perp$ (a contradiction), there exists a finite fragment $\Delta$ of ZF+¬AC such that $\Delta$ proves $\perp$. In other words, $\Delta$ is inconsistent. However, by Proposition 2.1.31, $\Delta$ is consistent, which leads to a contradiction. Therefore, ZF+¬AC is consistent.

## 2.2 Outline for the Informal Proof

We outline an informal proof of the relative consistency of ¬AC with ZF which we will formalize in the next chapter. In this proof, the relative consistency is proved by assuming the existence of a c.t.m. of ZF and constructing a model of ZF+¬AC by forcing. This model is a symmetric extension called the basic Cohen model.

[2] Since the Reflection Principle is a theorem schema, this is also a theorem schema for every $\Delta$.

Let $M$ be a c.t.m. of ZF, $\mathbb{P}$ be the set of all finite partial functions from $\omega \times \omega$ to $2 = \{0, 1\}$ that are elements of $M$, and $\leq_{\mathbb{P}}$ be $\supseteq$. Note that the maximum element $1_{\mathbb{P}}$ is the empty set. Let $\pi$ be a bijection on $\omega$. $\pi$ induces an automorphism on $\mathbb{P}$ defined as follows:

$$\mathrm{dom}(\pi p) = \{(\pi n, m) \mid (n, m) \in \mathrm{dom}(p)\}$$

$$(\pi p)(\pi n, m) = p(n, m)$$

This automorphism further induces an automorphism on $\mathbb{P}$-names. Let $\mathcal{G}$ be the group of all such automorphisms. For every finite $e \subseteq \omega$, let

$$\mathrm{fix}(e) = \{\pi \in \mathcal{G} \mid \forall n \in e(\pi n = n)\}$$

Let $\mathcal{F}$ be the set of all subgroups $H$ of $\mathcal{G}$ such that there exists a finite $e \subseteq \omega$ with $\mathrm{fix}(e) \subseteq H$. Note that $\mathcal{F}$ is a normal filter on $\mathcal{G}$. Let $\mathcal{N} = \mathrm{HS}^G$. Since $\mathcal{N}$ is a symmetric extension of $M$, it is a c.t.m. of ZF.

**Theorem 2.2.1.** *$\mathcal{N}$ does not satisfy the well-ordering theorem.*

*Proof.* We outline the proof of this theorem as follows. For every $n \in \omega$, let $a_n$ be the following real number (a subset of $\omega$):

$$a_n = \{m \in \omega \mid \exists p \in G(p(n, m) = 1)\}$$

Since $a_n$ are pairwise distinct, $A = \{a_n \mid n \in \omega\}$ is an infinity set. $A$ and every $a_n$ are elements of $\mathcal{N}$. $A$ serves as a counterexample to the well-ordering theorem in $\mathcal{N}$. Suppose for contradiction that $A$ is well-ordered in $\mathcal{N}$, there exists a injection $f$ from $\omega$ to $A$ in $\mathcal{N}$. Let $\varphi(g, x, y)$ be a formula that represents the relation $g(x) = y$. For every $n \in \omega$ with $a_n \in \mathrm{ran}(f)$, there exists $i \in \omega$ such that $N \vDash \varphi(f, i, a_n)$. Thus there exists $p \in G$ and hereditarily symmetric names $\dot{f}, \dot{i}$ and $\dot{a_n}$ for each of $f, i, a_n$ such that

$$p \Vdash_{\mathrm{HS}} \varphi(\dot{f}, \dot{i}, \dot{a_n})$$

By choosing $n$ and the names appropriately, we can find a bijection $\pi$ on $\omega$ such that the following conditions are additionally satisfied:

1. $\pi \dot{f} = \dot{f}$
2. $\pi \dot{i} = \dot{i}$
3. $\pi n \neq n$
4. There exists a hereditarily symmetric name $\dot{a_{\pi n}}$ of $a_{\pi n}$ such that $\pi \dot{a_n} = \dot{a_{\pi n}}$
5. There exists $q \in G$ such that $q \leq_{\mathbb{P}} p$ and $q \leq_{\mathbb{P}} \pi p$

Note that some occurence of $\pi$ in above conditions refer to the induced automorphism on $\mathbb{P}$ or $\mathbb{P}$-names.. By Lemma 2.1.24

$$\pi p \Vdash_{\mathrm{HS}} \varphi(\pi \dot{f}, \pi \dot{i}, \pi \dot{a_n})$$

Thus

$$\pi p \Vdash_{\mathrm{HS}} \varphi(\dot{f}, \dot{i}, \dot{a_{\pi n}})$$

Therefore, by Lemma 2.1.23

$$q \Vdash_{\mathrm{HS}} \varphi(\dot{f}, \dot{i}, \dot{a_n})\text{and}q \Vdash_{\mathrm{HS}} \varphi(\dot{f}, \dot{i}, \dot{a_{\pi n}})$$

This means that $\mathcal{N} \vDash \varphi(f, i, a_n)$ and $\mathcal{N} \vDash \varphi(f, i, a_{\pi n})$, which implies that $f(i) = a_n$ and $f(i) = a_{\pi n}$. Since $a_n$ and $a_{\pi n}$ are distinct, this is a contradiction. □

## 2.3 Isabelle/ZF and Formalization in Prior Work

In this section, we introduce Isabelle/ZF, a proof assistant for ZF set theory, and important results from prior work used in the formalization of this study.

### 2.3.1 Isabelle and Isabelle/ZF

Isabelle is a proof assistant developed [22], used for formalizing mathematical proofs and verifying programs. Since its release in 1986, it has continued to be developed by many researchers.

One notable example of Isabelle's achievements is the formal verification of the seL4 microkernel [15]. This was a massive project involving over a million lines of Isabelle code. More recently, in The ALEXANDRIA Project [21], various advanced mathematical results have been formalized in Isabelle.

Isabelle is a tool for proving theorems of the loical system called Pure. On top of Pure, other logical systems such as first-order logic (Isabelle/FOL) and Higher-Order Logic (Isabelle/HOL) are built, and Isabelle allows proving theorems in these logical systems.

Isabelle/ZF is an extension of Isabelle/FOL. It extends Isabelle/FOL by adding the axioms of ZF, allowing theorems of ZF set theory to be proved. Additionally, the Axiom of Choice can be included, enabling theorems of ZFC to be proved in Isabelle/ZF. Many definitions and theorems regarding ZF, such as ordinals, cardinals, functions, and ordered sets, have been formalized and provided as a library.

### 2.3.2 Important Results from Prior Work in Isabelle/ZF

#### Recursive Data Types and Functions

[20] introduced "datatype" to recursive data types within ZF set theory, and ”primrec” to define functions on top of these data types. This made it possible to define data structures such as lists within ZF set theory. For example, the set of all lists of elements from a set $A$ is defined in a library of Isabelle/ZF as follows:

```
consts
  list      :: "i⇒i"
```

```
datatype
  "list(A)" = Nil | Cons ("a ∈ A", "l ∈ list(A)")
```

Additionally, the function ”length” on lists is defined using ”primrec” as follows.

```
consts
  length :: "i⇒i"

primrec
  "length([]) = 0"
  "length(Cons(a,l)) = succ(length(l))"
```

### Class and Relativization

In Isabelle/ZF, a class $C$ is represented as a unary predicate $C(x)$. The statement $x \in C$ is represented as $C(x)$. For example, the class of all ordinals Ord is defined in a library of Isabelle/ZF as "Ord", and the fact that "α" is an ordinal is represented as "Ord(α)".

Additionally, the notation "∀x[C]. P(x)" represents that "P(x)" holds for all "x" in the class "C", and "∃x[C]. P(x)" represents that there exists an "x" in the class "C" such that "P(x)" holds. Using this notation, we can obtain the relativized statement to a class "C" by replacing all "∀x" and "∃x" in a original statement with "∀x[C]" and "∃x[C]", respectively.

### Internalized Formula and Satisfaction Relation

Paluson defined formulas encoded as sets in ZF set theory in their work [24], using "datatype" as follows:

```
consts  formula :: i
datatype
  "formula" = Member ("x ∈ nat", "y ∈ nat")
            | Equal  ("x ∈ nat", "y ∈ nat")
            | Nand ("p ∈ formula", "q ∈ formula")
            | Forall ("p ∈ formula")
```

Note that "nat" is the set of natural numbers. The unique logical connective in this definition is NAND and other logical connectives such as AND, OR, and NOT on formulas are defined in terms of NAND. Formulas are represented using the de Brujin index, in other words, a occurence of a natural number $n$ in a formula represents the $(n - m)$-th free variable when this $n$ occurs under $m$ quantifiers.

Additionally, the satisfaction relation "sats(A, φ, env)" is defined. This relation represents that "A" satisfies the formula "φ" under the environment "env", where "env" is a list of elements of "A". Note that the arity of "φ" must be less than or equal to the length of "env".

For example, "sats(A, Member(0, 1), [a, b])" is equivalent to "a ∈ b" when "[a, b] ∈ list(A)", and "sats(A, Forall(Member(0, 1)), [c])" is equivalent to "∀x ∈ A. c ∈ x" when "[c] ∈ list(A)".

### Locales and Internalized ZF Axioms

Isabelle provides a feature called locales, which allows us to work more easily in a specific context by fixing constants and introducing assumptions. For example, when formalizing group theory, rather than adding the assumption ”*G* is a group” to every theorem, we can simply declare a locale with the constant *G* and the assumption ”*G* is a group”.

Paulson and Gunther et al. declared locales to handle internal models of ZF set theory in their work [24, 8], respectively. For example, the locale "M_ctm" by Gunther et al. fixes "M" and "enum", and introduces assumptions that "M" holds the axioms of ZF set theory, "M" is a transitive set, and "enum" is a enumeration of "M". Therefore, we can work in the context of a c.t.m. of ZF by working within this locale.

### Forcing

Gunther et al. formalized forcing method in ZF set theory [8]. Their formalization follows Kunen’s book [16], which uses the c.t.m. approach.

They defined the notion of forcing as the locale "forcing_notion" with the constants "P", "leq", and "one".

```
locale forcing_notion =
  fixes P leq one
  assumes one_in_P:     "one ∈ P"
    and leq_preord:     "preorder_on(P,leq)"
    and one_max:        "∀p∈P. ⟨p,one⟩∈leq"
```

They also declared the locale "forcing_data" by combining the locale "forcing_notion" with the locale "M_ctm" and assume some additional properties.

```
locale forcing_data = forcing_notion + M_ctm +
  assumes P_in_M:        "P ∈ M"
    and leq_in_M:        "leq ∈ M"
```

Within this locale, The predicate "p ⊩ φ env" is defined. This represents that "p" forces the formula "φ" under the environment "env". Additionally, the following [3] is their formalization of Theorem 2.1.12 and Corollary 2.1.13.

```
lemma truth_lemma:
  assumes
    "φ∈formula" "M_generic(G)"
  shows
```

[3] In Definition 2.1.11, we explained that the forcing relation is defined for the formulas in the forcing language. According to that definition, "env" should be a list of elements of $M^{\mathbb{P}}$; however, in the following lemmas, "env" is a list of elements of $M$. In fact, the definition of the forcing relation by Gunther et al. differs from Definition 2.1.11. They defined the forcing relation not for the formulas in the forcing language but for the formulas in the extended language of set theory by adding all elements of $M$ as constants. Additionally, only NAND is used as the logical connective, and only ∀ is used as the quantifier of the formulas.

```
      "⋀env. env∈list(M) ⟹ arity(φ)≤length(env) ⟹
        (∃p∈G. p ⊩ φ env)   ⟷   M[G], map(val(G),env) ⊨ φ"

  lemma definition_of_forcing:
    assumes
      "p∈P" "φ∈formula" "env∈list(M)" "arity(φ)≤length(env)"
    shows
      "(p ⊩ φ env) ⟷
      (∀G. M_generic(G) ∧ p∈G  ⟶  M[G], map(val(G),env) ⊨ φ)"
```

Where "M_generic(G)" represents that "G" is a generic filter on "P" in "M", "val(G)" is the interpretation by "G", and "map" is the map function on lists.

# Chapter 3

# Formalization of the Proof

In this chapter, we present the formalization of the relative consistency proof of ¬AC with ZF in Isabelle/ZF. We follow the outline presented in Section 2.2. We choose *ZF-Constructible* by [24] as Isabelle theory [1] , which includes the ZF axioms along with useful definitions and lemmas built on top of them. We also reuse the formalization of forcing by [8].

## 3.1 Symmetric Extensions

### 3.1.1 Defining Symmetric Extensions

First, we define symmetric extensions in Isabelle/ZF, formalizing automorphisms, normal filters, and hereditarily symmetric names.

**$\mathbb{P}$-names**

We begin our work within the locale "forcing_data" by [8], in which a c.t.m. "M" and a notion of forcing "P" are fixed. The relation $\leq_{\mathbb{P}}$ is denoted by "leq" and a maximum element $1_{\mathbb{P}}$ is denoted by "one".

Gunther et al. did not provide an excplicit formalization of the set of $\mathbb{P}$-names $M^{\mathbb{P}}$ in their work. In this study, since we need to consider a subset of $M^{\mathbb{P}}$, we define $M^{\mathbb{P}}$ explicitly. We define $M^{\mathbb{P}}_{\alpha}$ as "P_set(α)" and $M^{\mathbb{P}}$ as "P_names" as follows:

https://github.com/tarakojo/ZF_notAC/blob/a376fdd7222b5b1fdb7681a1f542521e8805d82e/code/P_Names.thy#L10

```
definition HP_set_succ :: "[i, i] ⇒ i" where
  "HP_set_succ(a, X) ≡ Pow(X × P) ∩ M"

definition P_set :: "i ⇒ i" where
  "P_set(a) ≡ transrec2(a, 0, HP_set_succ)"

definition P_names :: "i" where "P_names ≡ { x ∈ M . ∃ a. Ord(a) ∧ x ∈ P_set(a) }"
```

Where "transrec2(α, x, H)" is the function defined in *ZF* library that returns x when α is 0; when α is a successor ordinal, it returns H(β, transrec2(β, x, H)) where β is the predecessor of α; and when α is a limit ordinal, it returns ⋃β∈α. transrec2(β, x, H). Note that in Isabelle/ZF, the notation "{ x ∈ A. P(x) }" denotes the set of elements in A that satisfy the predicate P.

[1] Note that this does not mean assuming $V = L$.

## Automorphisms

To simplify the formalization, we declare and work within the locale "forcing_data_partial", where ℙ is assumed to be a partially ordered set. Note that "one" is the unique maximum element of ℙ in this locale.

```
locale forcing_data_partial = forcing_data +
  assumes leq_relation_on_P : "leq ∈ Pow(P × P)"
  and leq_partial_order : "partial_order_on(P, leq)"
```

https://github.com/tarakojo/ZF_notAC/blob/a376fdd7222b5b1fdb7681a1f542521e8805d82e/code/Automorphism_Definition.thy#L5

We define the set of automorphisms on ℙ, denoted by "P_auto", as follows:

```
definition is_P_auto :: "i ⇒ o" where
  "is_P_auto(π) ≡ π ∈ M ∧ π ∈ bij(P, P) ∧ (∀ p ∈ P. ∀q ∈ P. p ≼ q ⟷ π`p ≼ π`q)"

definition P_auto where "P_auto ≡ { π ∈ P → P. is_P_auto(π) }"
```

Automorphism_Definition.thy:25-28

Where "bij(A, B)" is the set of bijections from A to B and "f`x" denotes the function application of f to x. We denote the induced automorphism on ℙ-names by π as "Pn_auto(π)". Our definition of "Pn_auto" satisfies the equality given in Definition 2.1.14.

```
lemma Pn_auto :
  "x ∈ P_names ⟹ Pn_auto(π)`x = { <Pn_auto(π)`y, π`p> . <y, p> ∈ x}"
```

## Groups of Automorphisms and Normal Filters

First, we define the set of all subgroups of a group of automorphisms G that are in "M".

```
 definition is_P_auto_group where
   "is_P_auto_group(G) ≡
     G ⊆ { π ∈ P → P. is_P_auto(π) }
   ∧ (∀π ∈ G. ∀τ ∈ G. π O τ ∈ G)
   ∧ (∀π ∈ G. converse(π) ∈ G)"

 definition P_auto_subgroups where
   "P_auto_subgroups(G) ≡ { H ∈ Pow(G) ∩ M. is_P_auto_group(H) }"
```

Where "π O τ" denotes the composition of π and τ. Then, we declare a locale "M_symmetric_system" that fixes a group of automorphisms "𝒢" and a normal filter "𝓕" on "𝒢".

```
locale M_symmetric_system = forcing_data_partial +
  fixes 𝒢 𝓕
  assumes 𝒢_in_M : "𝒢 ∈ M"
  and 𝒢_P_auto_group : "is_P_auto_group(𝒢)"
  and 𝓕_in_M : "𝓕 ∈ M"
  and 𝓕_subset : "𝓕 ⊆ P_auto_subgroups(𝒢)"
  and 𝓕_nonempty : "𝓕 ≠ 0"
  and 𝓕_closed_under_intersection : "∀A ∈ 𝓕. ∀B ∈ 𝓕. A ∩ B ∈ 𝓕"
  and 𝓕_closed_under_supergroup :
    "∀A ∈ 𝓕. ∀B ∈ P_auto_subgroups(𝒢). A ⊆ B ⟶ B ∈ 𝓕"
  and 𝓕_normal : "∀H ∈ 𝓕. ∀π ∈ 𝒢. { π O τ O converse(π). τ ∈ H } ∈ 𝓕"
```

## Hereditarily Symmetric Names

We continue our work within the locale "M_symmetric_system". We define the set of hereditarily symmetric names "HS" in the same way as "P_names". This approach makes it easier to use induction based on the ℙ-rank of names.

```
definition sym where "sym(x) ≡ { π ∈ 𝒢. Pn_auto(π)`x = x }"

definition symmetric where "symmetric(x) ≡ sym(x) ∈ 𝓕"

definition HHS_set_succ where
 "HHS_set_succ(a, X) ≡ { x ∈ P_set(succ(a)). domain(x) ⊆ X ∧ symmetric(x) }"

definition HS_set where "HS_set(a) ≡ transrec2(a, 0, HHS_set_succ)"

definition HS where "HS ≡ { x ∈ P_names. ∃a. Ord(a) ∧ x ∈ HS_set(a) }"
```

The following lemma shows our definition of "HS" is equivalent to Definition 2.1.17.

```
lemma HS_iff: "x ∈ HS ⟷ x ∈ P_names ∧ domain(x) ⊆ HS ∧ symmetric(x)"
```

## Symmetric Extensions

Finally, we define symmetric extensions.

```
definition SymExt where "SymExt(G) ≡ { val(G, x). x ∈ HS }"
```

Where "val(G, x)" is the interpretation $x^G$ formalized by [8].

We declare a locale "M_symmetric_system_G_generic", which combines the locale "M_symmetric_system" and the locale "G_generic". "G_generic" is a locale, defined by [8], fixes a generic filter "G" on "P".

```
locale M_symmetric_system_G_generic = M_symmetric_system + G_generic
```

The following lemmas can be proved relatively easily in this locale.

```
lemma M_subset_SymExt : "M ⊆ SymExt(G)"
lemma SymExt_subset_GenExt : "SymExt(G) ⊆ GenExt(G)"
lemma Transset_SymExt : "Transset(SymExt(G))"
lemma SymExt_countable : "nat ≈ SymExt(G)"
```

Where "GenExt(G)" is the generic extension generated by "G", "Transset(A)" denotes that "A" is transitive, and "nat ≈ A" denotes that "A" has the same cardinality as the set of natural numbers.

## Formulas Representing These Definitions

In our proofs, we need to use the separation and replacement axioms in M based on formulas that include the concepts defined in this section. Therefore, it is necessary to represent these concepts as internalized formulas.

We utilized the automated formula synthesis method by [8]; however, it could not applied to complex formulas, and re-verifying the properties of the derived formulas in a more usable form remained a repititive task.

Example of the formulas we defined includes "is_P_auto_group", "is_P_name_fm", "is_Pn_auto_fm", and "is_HS_fm", whick represent "P_names", "Pn_auto" and "HS" respectively. The correctness of these formulas is verified by proving the following lemmas.

```
lemma sats_is_P_name_fm_iff :
  fixes env i j x
  assumes "env ∈ list(M)" "i < length(env)" "j < length(env)"
          "nth(i, env) = P" "nth(j, env) = x"
  shows "sats(M, is_P_name_fm(i, j), env) ⟷ x ∈ P_names"

lemma sats_is_Pn_auto_fm_iff :
  fixes x π v env i j k l
  assumes "i < length(env)" "j < length(env)" "k < length(env)" "l < length(env)"
          "nth(i, env) = P" "nth(j, env) = π" "nth(k, env) = x" "nth(l, env) = v"
          "env ∈ list(M)" "π ∈ P_auto"
  shows "sats(M, is_Pn_auto_fm(i, j, k, l), env) ⟷ x ∈ P_names ∧ v = Pn_auto(π)`x"

lemma sats_is_HS_fm_iff :
  fixes x i j env
  assumes "env ∈ list(M)" "i < length(env)" "j < length(env)"
          "nth(i, env) = <ℱ, 𝒢, P, P_auto>" "nth(j, env) = x"
  shows "sats(M, is_HS_fm(i, j), env) ⟷ x ∈ HS"
```

Note that these formulas take as parameters various elements fixed within the locale, such as the notion of forcing "P", the normal filter "ℱ", etc.

# 3.1.2 Proving Symmetric Extensions are Models of ZF

In this section, we prove that symmetric extensions are models of ZF. Initially, we attempted to formalize this based on its proof in [17, Theorem 10.10], but we found that additional work was required to formalize parts that are omitted in the note. Therefore, we decided to prove this by using the relativized forcing relation $\Vdash_{\mathrm{HS}}$. Initial approach is a traditional method for proving that symmetric extensions are models of ZF, and applicable to a more general situation since it does not rely on forcing or generic and symmetric extensions. whereas our approach is more specific to the context of symmetric extensions.

## Traditional Approach

In [17, Theorem 10.10] and [13], the fact that symmetric extensions are models of ZF is proven by showing that they are internal models of the conrresponding generic extension generated by the same notion of forcing and generic filter, using the following proposition:

**Proposition 3.1.1.** *If $\mathcal{N}$ is a transitive class that is almost universal and satisfies $\Delta_0$-separation, then $\mathcal{N}$ is a inner model of ZF.*

Where a class $\mathcal{N}$ is said to be *almost universal* [2] if every subset $X \subseteq \mathcal{N}$ is included in some $Y \in \mathcal{N}$. Furthermore, $\mathcal{N}$ is said to satisfy $\Delta_0$-separation if it satisfies the separation axioms for formulas with only bounded quantifiers.

We could prove that the symmetric extension "SymExt(G)" is transitive, almost universal and satisfies $\Delta_0$-separation. However, we found that proving that "SymExt(G)" is actually a class in "GenExt(G)" is not straightforward in Isabelle/ZF. To do this, we needed to explicitly construct a formula that represents the set of all names "P_names" on the ground model "M" in "GenExt(G)". However, this required additional work, which would likely involve complex discussions. Therefore, we decided to adopt a different approach.

## Our Approach : The Relativized Forcing Relation $\Vdash_{\mathrm{HS}}$

Instead, we formalize the relativized forcing relation $\Vdash_{\mathrm{HS}}$ and prove that symmetric extensions are models of ZF using it. By modifying the formalization of forcing by [8], we were able to define this relation and prove its properties relatively easily.

Within the locale "M_symmetric_system", we define this relation in the form "p ⊩HS φ env", which has the same form as the formalization of the original forcing relation "p ⊩ φ env" by Gunther et al. Our definition satisfies the required properties, such as Theorem 2.1.21, Corollary 2.1.22, and Lemma 2.1.23.

[2] Jech [13], Theorem 13.9.

```
lemma HS_truth_lemma:
  assumes
    "φ∈formula" "M_generic(G)"
  shows
    "⋀env. env∈list(HS) ⟹ arity(φ)≤length(env) ⟹
       (∃p∈G. p ⊩HS φ env) ⟷ SymExt(G), map(val(G),env) ⊨ φ"

lemma definition_of_forcing_HS:
  assumes
    "p∈P" "φ∈formula" "env∈list(HS)" "arity(φ)≤length(env)"
  shows
    "(p ⊩HS φ env) ⟷
       (∀G. M_generic(G) ∧ p∈G  ⟶  SymExt(G), map(val(G),env) ⊨ φ)"

lemma HS_strengthening_lemma:
  assumes
    "p∈P" "φ∈formula" "r∈P" "r≼p"
  shows
    "⋀env. env∈list(M) ⟹ arity(φ)≤length(env) ⟹ p ⊩HS φ env ⟹ r ⊩HS φ env"
```

# The Symmetry Lemma

Additionally, we prove the symmetry lemma (Lemma 2.1.24). Similary to the definitions of the forcing relations, this lemma is proven using a complex induction. First, we prove this lemma for atomic formulas in the forcing language, and then extend the proof to other formulas by structual induction on the formula. For atomic formulas $\dot{x} = \dot{y}$ and $\dot{x} \in \dot{y}$, the definition is mutually recursive, so the lemma for these formulas is proven simultaneously by induction on the $\mathbb{P}$-rank of $\dot{x}$ and $\dot{y}$. Specifically, we show these induction steps:

```
 have MEM_step: "⋀a. Ord(a) ⟹ ∀b ∈ a. Q(EQ, b, b) ⟹ ∀b ∈ a. Q(MEM, b, a)"
 have EQ_step : "⋀a. Ord(a) ⟹ ∀b ∈ a. Q(MEM, b, a) ⟹ Q(EQ, a, a)"
```

Where "Q(MEM, α, β)" and "Q(EQ, α, β)" are predicates representing that the symmetriy lemma holds for $\dot{x} \in \dot{y}$ and $\dot{x} = \dot{y}$, respectively, for any $\dot{x}$ with $\mathbb{P}$-rank at most $\alpha$ and $\dot{y}$ with $\mathbb{P}$-rank at most $\beta$.

The proof for non-atomic formulas is relatively straightforward, and finally, this completes the proof of the lemma.

```
lemma symmetry_lemma:
  fixes φ π
  assumes "φ ∈ formula" "is_P_auto(π)" "π ∈ 𝒢"
  shows "⋀env p. env ∈ list(HS) ⟹ arity(φ) ≤ length(env) ⟹ p ∈ P ⟹
          p ⊩HS φ env ⟷ π`p ⊩HS φ map(λx. Pn_auto(π)`x, env)"
```

# Separation

Using the symmetry lemma, we prove the axiom schema of separation for symmetric extensions within the locale "M_symmetric_system_G_generic". Our proof is based on Karagila's proof [17] of $\Delta_0$-separation for symmetric extensions. However, since we use the $\Vdash_{\text{HS}}$ relation instead of $\Vdash$, we can prove separation for any formula, not just $\Delta_0$ formulas.

The separation for the formula $\varphi$ means that for any elements $x, p$ in the symmetric extension, the set $X = \{y \in x \mid \varphi(y, p)\}$ is also an element of it. To prove this, we show that $\dot{X} = \{\langle y, p\rangle \in \text{dom}(x) \times \mathbb{P} \mid p \Vdash_{\text{HS}} \varphi(\dot{y}, \dot{p})\}$ is a hereditarily symmetric name of $X$, i.e., $\dot{X} \in \text{HS}$ and $\dot{X}^G = X$.

```
lemma sep_forces_pair_in_HS :
  fixes x env φ
  assumes "x ∈ HS" "env ∈ list(HS)" "φ ∈ formula" "arity(φ) ≤ succ(length(env))"
  shows "{ <y, p> ∈ domain(x) × P. p ⊩HS φ [y] @ env } ∈ HS"

lemma SymExt_separation :
  fixes x env φ
  assumes "x ∈ SymExt(G)" "env ∈ list(SymExt(G))"
          "φ ∈ formula" "arity(φ) ≤ succ(length(env))"
  shows "{ y ∈ x. sats(SymExt(G), φ, [y] @ env) } ∈ SymExt(G)"
```

# Replacement

Now that we have proven separation, to prove the axiom schema of replacement for symmetric extensions, it is sufficient to show that for any formula $\varphi$ and any elements $x, p$ in a symmetric extension $\mathcal{N}$, we can find a set $S \in \mathcal{N}$ such that :

$$\mathcal{N} \vDash \exists z\, \varphi(y, z, p) \text{ iff } \mathcal{N} \vDash \exists z\, (z \in S \wedge \varphi(y, z, p)) \text{ for all } y \in x$$

This is because such a set $S$ contains the image of $x$ under the class function defined by $\varphi$ [3], which is the set that would be obtained by applying the replacement axiom. To prove this, we show the following lemma, which rewrites the above statement in terms of the forcing relation $\Vdash_{\text{HS}}$, and then prove it.

```
lemma ex_hs_subset_contains_witnesses :
  fixes φ env x
  assumes "φ ∈ formula" "env ∈ list(M)" "arity(φ) ≤ 2 #+ length(env)" "x ∈ M"
  shows "∃S. S ∈ M ∧ S ⊆ HS ∧ (∀p ∈ G. ∀y ∈ domain(x).
          (∃z ∈ HS. p ⊩HS φ ([y, z] @ env)) ⟷ (∃z ∈ S. p ⊩HS φ ([y, z] @ env)))"

 lemma ex_SymExt_elem_contains_witnesses :
  fixes φ env x
  assumes "φ ∈ formula" "env ∈ list(SymExt(G))"
          "arity(φ) ≤ 2 #+ length(env)" "x ∈ SymExt(G)"
  shows "∃S ∈ SymExt(G). ∀y ∈ x.
          ((∃z ∈ SymExt(G). sats(SymExt(G), φ, [y, z] @ env))
```

[3] If $\varphi$ defines a class function.

```
                ⟷ (∃z ∈ S. sats(SymExt(G), φ, [y, z] @ env)))"
```

Additionally, the following lemma was helpful in these proofs. This is also frequently used in the proofs of the axioms.

```
lemma ex_separation_base :
  fixes X
  assumes "X ⊆ HS" "X ∈ M"
  shows "∃S ∈ SymExt(G). { val(G, x). x ∈ X } ⊆ S"
```

This lemma states that for any set of hereditarily symmetric names $X \in M$, there exists a set $S$ in the symmetric extension such that $X^G \subseteq S$.

Then, we prove the axiom schema of replacement for symmetric extensions.

```
lemma SymExt_replacement :
  fixes φ env
  assumes "φ ∈ formula" "arity(φ) ≤ 2 #+ length(env)" "env ∈ list(SymExt(G))"
  shows
     "strong_replacement(##SymExt(G), λx y. sats(SymExt(G), φ, [x, y] @ env))"
```

#### Other Axioms

Other axioms of ZF can be proven relatively easily in symmetric extensions by using separation, replacement and the lemma "ex_separation_base". For example, the axiom of pairing can be proven as follows. Let a, b be elements of a symmetric extension $\mathcal{N}$. Then, we can take hereditarily symmetric names $\dot{a}, \dot{b}$ of a, b, respectively. By the lemma "ex_separation_base", we can find a set $S$ in $\mathcal{N}$ such that $\{\dot{a}, \dot{b}\}^G \subseteq S$. In other words, $\{a, b\} \subseteq S$. Then by the separation axiom, $\{x \in S \mid x = a \vee x = b\} = \{a, b\}$ is an element of $\mathcal{N}$.

Finally we obtain the following lemma, which states that symmetric extensions are models of ZF.

```
theorem SymExt_sats_ZF : "SymExt(G) ⊨ ZF"
```

Furthermore, since symmetric extensions are transitive, it can be shown that they are instances of the locale "M_ZF_trans". As a result, many results concerning ZF set theory formalized in this locale can now be applied to symmetric extensions.

```
lemma SymExt_M_ZF_trans : "M_ZF_trans(SymExt(G))"
```

## 3.2 The Basic Cohen Model

### 3.2.1 Defining the Basic Cohen Model

In this section, we define the basic Cohen model which is a symmetric extension that does not satisfy the Axiom of Choice, , as outlined in Section 2.2. First, we work within the locale "M_ctm" which fixes a c.t.m. "M" of ZF.

#### The Notion of Forcing

the notion of forcing for the basic Cohen model is defined as the set of finite partial functions from $\omega \times \omega$ to $2$ that are elements of the ground model, with the order relation given by the superset relation. We define these as "Fn" and "Fn_leq" in Isabelle/ZF.

```
definition Fn where "Fn ≡ { f ∈ Pow((nat × nat) × 2) ∩ M.
                function(f) ∧ domain(f) ⊆ nat × nat ∧ finite_M(domain(f)) ∧ range(f) ⊆ 2 }"

definition Fn_leq where "Fn_leq ≡ { <f, g> ∈ Fn × Fn. g ⊆ f }"
```

Where "nat" denotes $\omega$ and "finite_M(A)" denotes that the set "A" is finite. From the following lemma, this ordered set is an instance of the locale "forcing_data_partial", meaning that this is indeed a notion of forcing and forms a partially ordered set with the empty set as the maximum element.

```
lemma Fn_forcing_data_partial : "forcing_data_partial(Fn, Fn_leq, 0, M, enum)"
```

Where "enum" is the enumeration function of the set "M" fixed in the locale "M_ctm".

#### The Group of Automorphisms and the Normal Filter

Next, we define the group of automorphisms "Fn_perms" on the forcing notion "Fn".

```
definition nat_perms where "nat_perms ≡ bij(nat, nat) ∩ M"
definition Fn_perm where "Fn_perm(f, p) ≡ { <<f`n, m>, l> . <<n, m>, l> ∈ p }"
definition Fn_perm' where "Fn_perm'(f) ≡ { <p, Fn_perm(f, p)>. p ∈ Fn }"
definition Fn_perms where "Fn_perms ≡ { Fn_perm'(f). f ∈ nat_perms }"
```

Note that "Fn_perm'(f)" is the automorphism induced by the bijection "f" on $\omega$, and "Fn_perm(f, p)" is the value of the automorphism "Fn_perm'(f)" on the partial function "p". We prove that "Fn_perm'(f)" is indeed an automorphism on "Fn" and "Fn_perms" is a group of automorphisms on "Fn", i.e., closed under composition and the inverse operation, and contains the identity function.

```
lemma Fn_perm'_is_P_auto :
  fixes f
  assumes "f ∈ nat_perms"
  shows "forcing_data_partial.is_P_auto(Fn, Fn_leq, M, Fn_perm'(f))"

lemma Fn_perms_group :
  "forcing_data_partial.is_P_auto_group(Fn, Fn_leq, M, Fn_perms)"
```

Then, we define the normal filter "Fn_perms_filter" on the group of automorphisms "Fn_perms", and verify that our definition satisfies the prerequisites of the locale "M_symmetric_system". This completes both the definition of the basic Cohen model and the proof that it is a model of ZF.

```
definition Fn_perms_filter where "Fn_perms_filter ≡ {
        H ∈ forcing_data_partial.P_auto_subgroups(Fn, Fn_leq, M, Fn_perms).
        ∃E ∈ Pow(nat) ∩ M. finite_M(E) ∧ Fix(E) ⊆ H }"

lemma Fn_M_symmetric_system :
  "M_symmetric_system(Fn, Fn_leq, 0, M, enum, Fn_perms, Fn_perms_filter)"
```

### 3.2.2 Proving the Basic Cohen Model does not Satisfy AC

Finaly, we prove that the basic Cohen model does not satisfy the Axiom of Choice. To do this, we construct a set $A$ that cannot be well-ordered in it. We work within the locale "M_symmetric_system" with parameters, as defined above, fixed to generate the basic Cohen model.

We formalize $a_n$ and $A$, introduced in Section 2.2, as "binmap_row(G, n)" and "binmap(G)", respectively, for given generic filter "G".

```
definition binmap_row where
  "binmap_row(G, n) ≡ { m ∈ nat. ∃p ∈ G. p`<n, m> = 1 }"

definition binmap where "binmap(G) ≡ { binmap_row(G, n). n ∈ nat }"
```

Then, we prove that $a_n$ are pairwise disjoint, which implies that $A$ is an infinite set.

```
lemma binmap_row_distinct :
  fixes G n m
  assumes "M_generic(G)" "n ∈ nat" "m ∈ nat" "n ≠ m"
  shows "binmap_row(G, n) ≠ binmap_row(G, m)"
```

Using this lemma, we prove that $p \Vdash_{\text{HS}}$ ”$\dot{F}$ is a injection from $\dot{\omega}$ to $\dot{A}$” leads a contradiction for any $p$ ∈ "Fn" and $\dot{F}$ ∈ "HS". This means that there is no injection from $\omega$ to $A$ in the basic Cohen model.

```
lemma no_injection :
 fixes F' p0
 assumes "F' ∈ HS" "p0 ∈ Fn"
           "ForcesHS(p0, injection_fm(0, 1, 2), [check(nat), binmap', F'])"
 shows False
```

Where "ForcesHS(p, φ, env)" is another notation for "p ⊩HS φ env", and "check(nat)" and "binmap'" are the hereditarily symmetric names of "nat" and "binmap", respectively.

Since $A$ is an infinite set, there exists an injection from $\omega$ to $A$. Therefore, using above lemma, we can show that assuming $A$ is well-orderable leads to a contradiction,

```
lemma no_wellorder :
  fixes r G
  assumes
     "M_generic(G)" "wellordered(##SymExt(G), binmap(G), r)" "r ∈ SymExt(G)"
  shows False
```

Where "wellordered(C, A, r)" denotes that the relation "r" is a well-ordering of the set "A" in the class "C", and "##" is an operator that converts a set to a class. These are defined by [24]. Specifically, the definition of "wellordered" is as follows [4]:

```
definition transitive_rel :: "[i⇒o,i,i]⇒o" where
  "transitive_rel(M,A,r) ≡
    ∀x[M]. x∈A ⟶ (∀y[M]. y∈A ⟶ (∀z[M]. z∈A ⟶
              ⟨x,y⟩∈r ⟶ ⟨y,z⟩∈r ⟶ ⟨x,z⟩∈r))"
definition linear_rel :: "[i⇒o,i,i]⇒o" where
  "linear_rel(M,A,r) ≡
    ∀x[M]. x∈A ⟶ (∀y[M]. y∈A ⟶ ⟨x,y⟩∈r | x=y | ⟨y,x⟩∈r)"

definition wellfounded_on :: "[i⇒o,i,i]⇒o" where
  — ‹every non-empty SUBSET OF ‹A› has an ‹r›-minimal element›
  "wellfounded_on(M,A,r) ≡
    ∀x[M]. x≠0 ⟶ x⊆A ⟶ (∃y[M]. y∈x ∧ ¬(∃z[M]. z∈x ∧ ⟨z,y⟩ ∈ r))"

definition wellordered :: "[i⇒o,i,i]⇒o" where
  — ‹linear and wellfounded on ‹A››
  "wellordered(M,A,r) ≡
    transitive_rel(M,A,r) ∧ linear_rel(M,A,r) ∧ wellfounded_on(M,A,r)"
```

[4] Note that we can show that if "wellordered(N, A, r)" holds for an element "A" in a class "N", which is a model of ZF, then "r" is irreflexive and assymetric, i.e., a strict total order.

Thus, the statement ”a set $\mathcal{N}$ satisfies well-ordering theorem” can be written as follows:

```
"∀A ∈ N. ∃r ∈ N. wellordered(##N, A, r)"
```

Using the above, we prove the following theorem and complete our relative consistency proof. This theorem has no additional assumptions from any locale.

```
theorem ZF_notAC_main_theorem :
  fixes M
  assumes "nat ≈ M" "M ⊨ ZF" "Transset(M)"
  shows "∃N. nat ≈ N ∧ N ⊨ ZF ∧ Transset(N)
          ∧ ¬(∀A ∈ N. ∃r ∈ N. wellordered(##N, A, r))"
```

This theorem states that there exists a c.t.m. "N" of ZF+¬AC if there exists a c.t.m. "M" of ZF.

# Chapter 4

# Conclusion

We formalized the relative consistency proof of ¬AC with ZF on the basis of the work by [24] and [8]. Specifically, assuming the existence of a c.t.m. of ZF, we constructed a model of ZF+¬AC known as the Basic Cohen model by forcing. To achieve this, we formalized the definition of symmetric extensions and the related lemmas.

Our formalization comprises approximately 15,000 lines of code, distributed roughly as follows:

- Definition of symmetric extensions: 3,000 lines
- Proof of symmetric extensions are models of ZF: 5,000 lines
- Definition of the basic Cohen model: 2,000 lines
- Proof of the basic Cohen model does not satisfy AC: 2,000 lines
- Other lemmas : 3,000 lines

This is as large as Paluson's formalization of the proof of relative consistency of AC (12,000 lines), Gunther et al.'s formalization of coercion (16,000 lines), and their proof of independence of CH (12,000 lines).

Through the formalization, we felt particularly deficient of functionality to efficiently formalize "object-level" arguments in axiomatic set theory.

In our proof, we needed to construct internalized formulas representing most of the concepts that belong to the ground model. This was to verify that the defined sets truly belong to the ground model and to enable application of the separation and replacement axioms of the ground model for formulas that include these concepts.

For constructing these formulas we utilized the sythesis method by [8]; however, this method could not be applied to complex formulas. When applied, we also needed to reprove properties of derived formulas in a more usable form.

We also needed to prove that the arity of each defined formula is below its respective threshold. For complex formulas, Isabelle's automated proof tools were not applicable, and we had to prove these properties manually.

Since these works are repetitive, further automation is desirable. Ideally, if the arguments of the base model could be handled without explicitly dealing with internalized formulas, the amount of work could be reduced when formalizing proofs by using an approach similar to ours.

Additionally, we used the c.t.m. approach, and the formalized argument only extends to the construction of a model of ZF+¬AC, assuming the existence of a c.t.m. of ZF. We did not formalize the justification of the c.t.m. approach itself. However, it is also desirable to formalize this part.

The main obstacle to formalizing this part lies in the reflection principle. The reflection principle for propositions of Isabelle was proved by [23]; however, since we are dealing with propositions formalized in the form of internalized formulas, we need a reflection principle for such formulas.

There are several approaches to address this. One approach is to use metaprogramming, which involves writing a program to automatically generate proofs for specific instances of the reflection principle. In this case, only a finite number of instances of the reflection principle can be utilized, and the finite set of logical formulas as parameters for each instance must be explicitly constructed. This means that the reflection principle cannot be applied to finite sets of logical formulas obtained non-constructively.

Another approach is to prove a general reflection principle for internalized formulas. This would require discussing "object-level within object-level", that is, a further layer of encoded set theory within models of ZF or ZFC. Since even "object-level" arguments, such as those handled in this formalization, are already sufficiently complex, this approach would be difficult unless the aforementioned automation is achieved.

The final approach is to use higher-order logic. In higher-order logic, it would be possible to prove the reflection principle for internalized formulas without engaging in the complex discussions of "object-level within object-level". However, this would require using a proof assistant other than Isabelle/ZF. Since the accumulated definitions and lemmas in Isabelle/ZF would not be available, it would be necessary to re-formalize set-theoretic concepts such as forcing.

## Future Work

The automation of "object-level" arguments and the formalization of the validity of the c.t.m. approach mentioned above are interesting future work. Additionally, it would be interesting to formalize more advanced topics on symmetric extensions, as seen in the latter parts of [17]. Furthermore, formalizing the fact that the consistency of an axiomatic system $S$ does not imply the existence of a c.t.m. of $S$, as well as alternative approaches to forcing, such as those using modal logic, and formalizing the connections between different approaches to forcing, are also intriguing topics for future exploration.

# Bibliography

[1] José Bacelar Almeida et al. "A Machine-Checked Proof of Security for AWS Key Management Service". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 63–78. ISBN: 9781450367479. DOI: 10.1145/3319535.3354228. URL: https://doi.org/10.1145/3319535.3354228.

[2] Thomas Braibant. "Coquet: A Coq Library for Verifying Hardware". In: *Certified Programs and Proofs*. Ed. by Jean-Pierre Jouannaud and Zhong Shao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 330–345. ISBN: 978-3-642-25379-9.

[3] Paul J. Cohen. "The Independence of the Continuum Hypothesis". In: *Proceedings of the National Academy of Sciences of the United States of America* 50.6 (1963), pp. 1143–1148. ISSN: 00278424, 10916490. URL: http://www.jstor.org/stable/71858 (visited on 12/10/2024).

[4] Lean Community. *Lean*. https://leanprover.github.io/. Accessed: 2024-12-23. 2015-2024.

[5] Solange Coupet-Grimal and Line Jakubiec. "Coq and hardware verification: A case study". In: *Theorem Proving in Higher Order Logics*. Ed. by Gerhard Goos et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 125–139. ISBN: 978-3-540-70641-0.

[6] Denis Firsov and Dominique Unruh. *Zero-Knowledge in EasyCrypt*. Cryptology ePrint Archive, Paper 2022/926. 2022. URL: https://eprint.iacr.org/2022/926.

[7] Georges Gonthier. "The Four Colour Theorem: Engineering of a Formal Proof". In: *Computer Mathematics*. Ed. by Deepak Kapur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 333–333. ISBN: 978-3-540-87827-8.

[8] Emmanuel Gunther, Miguel Pagano, and Pedro Sánchez Terraf. "Formalization of Forcing in Isabelle/ZF". In: *Archive of Formal Proofs* (May 2020). https://isa-afp.org/entries/Forcing.html, Formal proof development. ISSN: 2150-914x.

[9] Emmanuel Gunther et al. "The Independence of the Continuum Hypothesis in Isabelle/ZF". In: *Archive of Formal Proofs* (Mar. 2022). https://isa-afp.org/entries/Independence_CH.html, Formal proof development. ISSN: 2150-914x.

[10] Thomas Hales et al. *A formal proof of the Kepler conjecture*. 2015. arXiv: 1501.02155 [math.MG]. URL: https://arxiv.org/abs/1501.02155.

[11] Jesse Michael Han and Floris van Doorn. "A formal proof of the independence of the continuum hypothesis". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*. 2020.

[12] M. Randall Holmes and Sky Wilshaw. *NF is Consistent*. 2024. eprint: arXiv : 1503.01406.

[13] Thomas J Jech. *Set Theory*. en. 3rd ed. Springer Monographs in Mathematics. Berlin, Germany: Springer, Oct. 2002.

[14] Thomas J Jech. *The axiom of choice*. Dover Books on Mathematics. Mineola, NY: Dover Publications, July 2008.

[15] Gerwin Klein, Philip Derrin, and Kevin Elphinstone. "Experience report: seL4: formally verifying a high-performance microkernel". In: *SIGPLAN Not.* 44.9 (Aug. 2009), pp. 91–96. ISSN: 0362-1340. DOI: 10.1145/1631687.1596566. URL: https://doi.org/10.1145/1631687.1596566.

[16] Kenneth Kunen. *Set Theory*. London, England: College Publications, Nov. 2011.

[17] *Lecture Notes: Forcing & Symmetric Extensions*. Accessed: 2024-10-29. 2023. URL: https://karagila.org/files/Forcing-2023.pdf.

[18] Xavier Leroy. "Formal verification of a realistic compiler". In: *Commun. ACM* 52.7 (July 2009), pp. 107–115. ISSN: 0001-0782. DOI: 10.1145/1538788.1538814. URL: https://doi.org/10.1145/1538788.1538814.

[19] M. Saqib Nawaz et al. "A Survey on Theorem Provers in Formal Methods". In: *CoRR* abs/1912.03028 (2019). arXiv: 1912.03028. URL: http://arxiv.org/abs/1912.03028.

[20] Lawrence C. Paulson. "A fixedpoint approach to implementing (Co)inductive definitions". In: *Automated Deduction — CADE-12*. Ed. by Alan Bundy. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 148–161. ISBN: 978-3-540-48467-7.

[21] Lawrence C. Paulson. "Large-Scale Formal Proof for the Working Mathematician—Lessons Learnt from the ALEXANDRIA Project". In: *Intelligent Computer Mathematics*. Ed. by Catherine Dubois and Manfred Kerber. Cham: Springer Nature Switzerland, 2023, pp. 3–15. ISBN: 978-3-031-42753-4.

[22] Lawrence C. Paulson. "Natural deduction as higher-order resolution". In: *The Journal of Logic Programming* 3.3 (1986), pp. 237–258. ISSN: 0743-1066. DOI: https://doi.org/10.1016/0743-1066(86)90015-4. URL: https://www.sciencedirect.com/science/article/pii/0743106686900154.

[23] Lawrence C. Paulson. "The Reflection Theorem: A Study in Meta-theoretic Reasoning". In: *Automated Deduction—CADE-18*. Springer Berlin Heidelberg, 2002, pp. 377–391. ISBN: 9783540456209. DOI: 10.1007/3-540-45620-1_31. URL: http://dx.doi.org/10.1007/3-540-45620-1_31.

[24] Lawrence C. Paulson. "The Relative Consistency of the Axiom of Choice — Mechanized Using Isabelle/ZF". In: *Logic and Theory of Algorithms*. Springer Berlin Heidelberg, pp. 486–490. ISBN: 9783540694076. DOI: 10.1007/978-3-540-69407-6_52. URL: http://dx.doi.org/10.1007/978-3-540-69407-6_52.

[25] LawrenceC. Paulson and Krzysztof Grabczewski. "Mechanizing set theory: Cardinal arithmetic and the Axiom of Choice". In: *Journal of Automated Reasoning* 17.3 (Dec. 1996). ISSN: 1573-0670. DOI: 10.1007/bf00283132. URL: http://dx.doi.org/10.1007/BF00283132.

[26] Talia Ringer et al. "QED at Large: A Survey of Engineering of Formally Verified Software". In: *Found. Trends Program. Lang.* 5.2–3 (Sept. 2019), pp. 102–281. ISSN: 2325-1107. DOI: 10.1561/2500000045. URL: https://doi.org/10.1561/2500000045.

[27] Agda Development Team. *Agda*. https://wiki.portal.chalmers.se/agda/. Accessed: 2024-12-23. 2005-2024.

[28] HOL Light Development Team. *HOL Light*. https://www.cl.cam.ac.uk/~jrh13/hol-light/. Accessed: 2024-12-23. 1996-2024.

[29] Isabelle Development Team. *Isabelle*. https://isabelle.in.tum.de/. Accessed: 2024-12-23. 1994-2024.

[30] The Coq Development Team. *The Coq Proof Assistant*. https://coq.inria.fr/. Accessed: 2024-12-23. 1989-2024.