Formalization of Forcing in Isabelle/ZF

Emmanuel Gunther[1], Miguel Pagano[1], and Pedro Sánchez Terraf[1,2(✉)]

[1] Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba, Córdoba, Argentina
{gunther,pagano,sterraf}@famaf.unc.edu.ar
[2] Centro de Investigación y Estudios de Matemática (CIEM-FaMAF), Conicet, Córdoba, Argentina

Abstract. We formalize the theory of forcing in the set theory framework of Isabelle/ZF. Under the assumption of the existence of a countable transitive model of *ZFC*, we construct a proper generic extension and show that the latter also satisfies *ZFC*. In doing so, we remodularized Paulson's `ZF-Constructibility` library.

Keywords: Forcing · Isabelle/ZF · Countable transitive models · Absoluteness · Generic extension · Constructibility

1 Introduction

The present work reports on the third stage of our project of formalizing the theory of forcing and its applications as presented in one of the more important references on the subject, Kunen's Set Theory [9] (a rewrite of the classical book [8]).

We work using the implementation of Zermelo-Fraenkel (*ZF*) set theory *Isabelle/ZF* by Paulson and Grabczewski [17]. In an early paper [3], we set up the first elements of the countable transitive model (ctm) approach, defining forcing notions, names, generic extensions, and showing the existence of generic filters via the Rasiowa-Sikorski lemma (RSL). In our second (unpublished) technical report [4] we advanced by presenting the first accurate *formal abstract* of the Fundamental Theorems of Forcing, and using them to show that the *ZF* axioms apart from Replacement and Infinity hold in all generic extensions.

This paper contains the proof of Fundamental Theorems and complete proofs of the Axioms of Infinity, Replacement, and Choice in all generic extensions. In particular, we were able to fulfill the promised formal abstract for the Forcing Theorems almost to the letter. A requirement for Infinity and the absoluteness of forcing for atomic formulas, we finished the interface between our development and Paulson's constructibility library [15] which enables us to do well-founded recursion inside transitive models of an appropriate finite fragment of *ZF*. As a by-product, we finally met two long-standing goals: the fact that the generic filter

Supported by Secyt-UNC project 33620180100465CB and Conicet.

N. Peltier and V. Sofronie-Stokkermans (Eds.): IJCAR 2020, LNAI 12167, pp. 221–235, 2020.
https://doi.org/10.1007/978-3-030-51054-1_13

G belongs to the extension $M[G]$ and $M \subseteq M[G]$. In order to take full advantage of the constructibility library we enhanced it by weakening the assumption of many results and also extended it with stronger results. Finally, our development is now independent of AC: We modularized RSL in such a way that a version for countable posets does not require choice.

In the course of our work we found it useful to develop Isar methods to automate repetitive tasks. Part of the interface with Paulson's library consisted in constructing formulas for each relativized concept; and actually Isabelle's Simplifier can synthesize terms for unbound schematic variables in theorems. The synthesized term, however, is not available outside the theorem; we introduced a method that creates a definition from a schematic goal. The second method is concerned with renaming of formulas: we improved our small library of bounded renamings with a method that given the source and target environments figures out the renaming function and produces the relevant lemmas about it.

The source code of our formalization, written for the 2019 version of Isabelle, can be browsed and downloaded at https://cs.famaf.unc.edu.ar/~pedro/forcing/.

We assume some familiarity with Isabelle and some terminology of set theory. The current paper is organized as follows. In Sect. 2 we comment briefly on the meta-theoretical implications of using Isabelle/ZF. In Sect. 3 we explain the use of relativized concepts and its importance for the ctm approach. The next sections cover the core of this report: In Sect. 4 we introduce the definition of the formula transformer `forces` and reasoning principles about it; in Sect. 5 we present the proofs of the fundamental theorems of forcing. We show in Sect. 6 a concrete poset that leads to a proper extension of the ground model. In Sect. 7 we complete the proof that every axiom and axiom scheme of ZFC is valid in any generic extension. Section 8 briefly discusses related works and we close the paper by noting the next steps in our project and drawing conclusions from this formalization.

2 Isabelle and (Meta)theories

Isabelle [14,18] is a general proof assistant based on fragment of higher-order logic called *Pure*. The results presented in this work are theorems of a version of *ZF* set theory (without the Axiom of Choice, AC) called *Isabelle/ZF*, which is one of the "object logics" that can be defined on top of Pure (which is then used as a language to define rules). Isabelle/ZF defines types `i` and `o` for sets and Booleans, resp., and the *ZF* axioms are written down as terms of type `o`.

More specifically, our results work under the hypothesis of the existence of a ctm of *ZFC*.[1] This hypothesis follows, for instance, from the existence of an inaccessible cardinal. As such, our framework is weaker than those found

[1] By Gödel's Second incompleteness theorem, one must assume at least the existence of some model of *ZF*. The countability is only used to prove the existence of generic filters and can be thus replaced in favor of this hypothesis.

usually in type theories with universes, but allows us to work "Platonistically"—assuming we are in a universe of sets (namely, `i`) and performing constructions there.

On the downside, our approach is not able to provide us with finitary consistency proofs. It is well known that, for example, the implication $\mathrm{Con}(ZF) \implies \mathrm{Con}(ZFC + \neg CH)$ can be proved in *primitive recursive arithmetic (PRA)*. To achieve this, however, it would have implied to work focusing on the proof mechanisms and distracting us from our main goal, that is, formalize the ctm approach currently used by many mathematicians.

It should be noted that Pure is a very weak framework and has no induction/recursion capabilities. So the only way to define functions by recursion is inside the object logic. (This works the same for Isabelle/HOL.) For this reason, to define the relation of forcing, we needed to resort to *internalized* first-order formulas: they form a recursively defined set `formula`. For example, the predicate of satisfaction `sats::i⇒i⇒i⇒o` (written $M, ms \models \varphi$ for a set M, $ms \in \mathtt{list}(M)$ and $\varphi \in \mathtt{formula}$) had already been defined by recursion in Paulson [15].

3 Relativization, Absoluteness, and the Axioms

The concepts of relativization and absoluteness (due to Gödel, in his proof of the relative consistency of AC [2]) are both prerequisites and powerful tools in working with transitive models. A *class* is simply a predicate $C(x)$ with at least one free variable x. The *relativization* $\varphi^C(\bar{x})$ of a set-theoretic definition φ (of a relation such as "x is a subset of y" or of a function like $y = \mathcal{P}(x)$) to a class C is obtained by restricting all of its quantifiers to C.

$$x \subseteq^C y \equiv \forall z.\ C(z) \longrightarrow (z \in x \longrightarrow z \in y)$$

The new formula $\varphi^C(\bar{x})$ corresponds to what is obtained by defining the concept "inside" C. In fact, for a class corresponding to a set c (i.e. $C(x) := x \in c$), the relativization φ^C of a sentence φ is equivalent to the satisfaction of φ in the first-order model $\langle c, \in \rangle$.

It turns out that many concepts mean the same after relativization to a nonempty transitive class C; formally

$$\forall \bar{x}.\ C(\bar{x}) \longrightarrow (\varphi^C(\bar{x}) \longleftrightarrow \varphi(\bar{x}))$$

When this is the case, we say that the relation defined by φ is *absolute for transitive models*.[2] As examples, the relation of inclusion $\subseteq$—and actually, any relation defined by a formula (equivalent to one) using only bounded quantifiers $(\forall x \in y)$ and $(\exists x \in y)$—is absolute for transitive models. On the contrary, this is not the case with the powerset operation.

A benefit of working with transitive models is that many concepts (pairs, unions, and fundamentally ordinals) are uniform across the universe `i`, a ctm

[2] Absoluteness of functions also requires the relativized definition to behave functionally over C.

(of an adequate fragment of ZF) M and any of its extensions $M[G]$. For that reason, then one can reason "externally" about absolute concepts, instead of "inside" the model; thus, one has at hand any already proved lemma about the real concept.

Paulson's formalization [15] of the relative consistency of AC by Gödel [2] already contains absoluteness results which were crucial to our project; we realized however that they could be refactored to be more useful. The main objective is to maximize applicability of the relativization machinery by adjusting the hypothesis of a greater part of its earlier development. Paulson's architecture had only in mind the consistency of ZFC, but, for instance, in order to apply it in the development of forcing, too much is assumed at the beginning; more seriously, some assumptions cannot be regarded as "first-order" (v.g. the Replacement Scheme) because of the occurrence of predicate variables. The version we present[3] of the constructibility library, `ZF-Constructible-Trans`, weakens the assumptions of many absoluteness theorems to that of a nonempty transitive class; we also include some stronger results such as the relativization of powersets.

Apart from the axiom schemes, the ZFC axioms are initially stated as predicates on classes (that is, of type `(i⇒o)⇒o`); this formulation allows a better interaction with `ZF-Constructible`. The axioms of Pairing, Union, Foundation, Extensionality, and Infinity are relativizations of the respective traditional first-order sentences to the class argument. For the Axiom of Choice we selected a version best suited for the work with transitive models: the relativization of a sentence stating that for every x there is surjection from an ordinal onto x. Finally, Separation and Replacement were treated separately to effectively obtain first-order versions afterwards. It is to be noted that predicates in Isabelle/ZF are akin to second order variables and thus do not correspond to first-order formulas. For that reason, Separation and Replacement are defined in terms of *the satisfaction* of an internalized formula φ. We improved their specification, with respect to our previous report [4], by lifting the arity restriction for the parameter φ; consequently we get rid of tupling and thus various proofs are now slicker.

A benefit of having class versions of the axioms is that we can apply our synthesis method to obtain their internal, first-order counterparts. For the case of the Pairing Axiom, the statement for classes is the following

```
upair_ax(C)==∀x[C]. ∀y[C]. ∃z[C]. upair(C,x,y,z)
```

where `upair` says that `z` is the unordered pair of `x` and `y`, relative to `C`.

The following schematic lemma synthesizes its internal version,[4]

schematic_goal *ZF_pairing_auto:*

[3] While preparing the final version of the present paper, our contributions were accepted as part of the official Isabelle 2020 release.

[4] The use of such schematic goals and the original definition of the collection of lemmas `sep_rules` are due to Paulson.

```
    "upair_ax(##A) ⟷ (A, [] ⊨ ?zfpair)"
unfolding upair_ax_def
  by ((rule sep_rules | simp)+)
```

and our `synthesize` method introduces a new term `ZF_pairing_fm` for it:

```
synthesize "ZF_pairing_fm" from_schematic "ZF_pairing_auto"
```

the actual formula obtained is `Forall(Forall(Exists(upair_fm(2,1,0))))`.

4 The Definition of `forces`

The core of the development is showing the definability of the relation of forcing. As we explained in our previous report [4, Sect. 8], this comprises the definition of a function `forces` that maps the set of internal formulas into itself. It is the very reason of applicability of forcing that the satisfaction of a first-order formula φ in all of the generic extensions of a ctm M can be "controlled" in a definable way from M (viz., by satisfaction of the formula $\texttt{forces}(\varphi)$).

In fact, given a forcing notion $\mathbb{P}$ (i.e. a preorder with a top element) in a ctm M, Kunen defines the *forcing relation* model-theoretically by considering all extensions $M[G]$ with G generic for $\mathbb{P}$. Then two fundamental results are proved, the Truth Lemma and the Definability Lemma; but the proof of the first one is based on the formula that witnesses Definability. To make sense of this in our formalization, we started with the internalized relation and then proved that it is equivalent to the semantic version ("`definition_of_forcing`," in the next section). For that reason, the usual notation of the forcing relation $p \Vdash \varphi\ env$ (for *env* a list of "names"), abbreviates in our code the satisfaction by M of $\texttt{forces}(\varphi)$:

```
"p ⊩ φ env    ≡    M, ([p,P,leq,one] @ env) ⊨ forces(φ)"
```

The definition of `forces` proceeds by recursion over the set `formula` and its base case, that is, for atomic formulas, is (in)famously the most complicated one. Actually, newcomers can be puzzled by the fact that forcing for atomic formulas is also defined by (mutual) recursion: to know if $\tau_1 \in \tau_2$ is forced by p (notation: $\texttt{forces\_mem}(p, \tau_1, \tau_2)$), one must check if $\tau_1 = \sigma$ is forced for σ moving in the transitive closure of τ_2. To disentangle this, one must realize that this last recursion must be described syntactically: the definition of $\texttt{forces}(\varphi)$ for atomic φ is then an internal definition of the alleged recursion on names.

Our aim was to follow the definition proposed by Kunen in [9, p. 257], where the following mutual recursion is given:

$$\texttt{forces\_eq}(p, t_1, t_2) := \forall s \in \mathrm{domain}(t_1) \cup \mathrm{domain}(t_2).\ \forall q \preceq p.\ \texttt{forces\_mem}(q, s, t_1) \longleftrightarrow \texttt{forces\_mem}(q, s, t_2), \quad (1)$$

$$\texttt{forces\_mem}(p, t_1, t_2) := \forall v \preceq p.\ \exists q \preceq v.\ \exists s.\ \exists r \in \mathbb{P}.\ \langle s, r\rangle \in t_2 \wedge q \preceq r \wedge \texttt{forces\_eq}(q, t_1, s) \quad (2)$$

Note that the definition of `forces_mem` is equivalent to require the set

$$\{q \preceq p : \exists \langle s, r\rangle \in t_2.\ q \preceq r \wedge \texttt{forces\_eq}(q, t_1, s)\}$$

to be dense below p.

It was not straightforward to use the recursion machinery of Isabelle/ZF to define `forces_eq` and `forces_mem`. For this, we defined a relation `frecR` on 4-tuples of elements of M, proved that it is well-founded and, more important, we also proved an induction principle for this relation:[5]

lemma *forces_induction:*
assumes
"⋀τ ϑ. ⟦⋀σ. σ∈domain(ϑ) ⟹ Q(τ,σ)⟧ ⟹ R(τ,ϑ)"
"⋀τ ϑ. ⟦⋀σ. σ∈domain(τ) ∪ domain(ϑ) ⟹ R(σ,τ) ∧ R(σ,ϑ)⟧
⟹ Q(τ,ϑ)"
shows
"Q(τ,ϑ) ∧ R(τ,ϑ)"

and obtained both functions as cases of a another one, `forces_at`, using a single recursion on `frecR`. Then we obtained (1) and (2) as our corollaries `def_forces_eq` and `def_forces_mem`.

Other approaches, like the one in Neeman [11] (and Kunen's older book [8]), prefer to have a single, more complicated, definition by simple recursion for `forces_eq` and then define `forces_mem` explicitly. On hindsight, this might have been a little simpler to do, but we preferred to be as faithful to the text as possible concerning this point.

Once `forces_at` and its relativized version `is_forces_at` were defined, we proceeded to show absoluteness and provided internal definitions for the recursion on names using results in `ZF-Constructible`. This finished the atomic case of the formula-transformer `forces`. The characterization of `forces` for negated and universal quantified formulas is given by the following lemmas, respectively:

lemma *sats_forces_Neg:*
assumes
"p∈P" "env ∈ list(M)" "φ∈formula"
shows
"M, [p,P,leq,one] @ env ⊨ forces(Neg(φ)) ⟷
¬(∃q∈M. q∈P ∧ is_leq(##M,leq,q,p) ∧
M, [q,P,leq,one]@env ⊨ forces(φ))"

lemma *sats_forces_Forall:*
assumes
"p∈P" "env ∈ list(M)" "φ∈formula"
shows
"M,[p,P,leq,one] @ env ⊨ forces(Forall(φ)) ⟷
(∀x∈M. M, [p,P,leq,one,x] @ env ⊨ forces(φ))"

[5] The logical primitives of *Pure* are ⟹, `&&&`, and ⋀ (implication, conjunction, and universal quantification, resp.), which operate on the meta-Booleans `prop`.

Let us note in passing another improvement over our previous report: we made a couple of new technical results concerning recursive definitions. Paulson proved absoluteness of functions defined by well-founded recursion over a transitive relation. Some of our definitions by recursion (*check* and *forces*) do not fit in that scheme. One can replace the relation R for its transitive closure R^+ in the recursive definition because one can prove, in general, that $F \restriction (R^{-1}(x))(y) = F \restriction \big((R^+)^{-1}(x)\big)(y)$ whenever $(x, y) \in R$.

5 The Forcing Theorems

After the definition of `forces` is complete, the proof of the Fundamental Theorems of Forcing is comparatively straightforward, and we were able to follow Kunen very closely. The more involved points of this part of the development were those where we needed to prove that various (dense) subsets of $\mathbb{P}$ were in M; for this, we have resorted to several ad-hoc absoluteness lemmas.

The first results concern characterizations of the forcing relation. Two of them are `Forces_Member`:

$$\texttt{(p} \Vdash \texttt{Member(n,m) env)} \longleftrightarrow \texttt{forces\_mem(p,t1,t2)},$$

where `t1` and `t1` are the `n`th resp. `m`th elements of `env`, and `Forces_Forall`:

$$\texttt{(p} \Vdash \texttt{Forall}(\varphi)\ \texttt{env)} \longleftrightarrow (\forall \texttt{x} \in \texttt{M}.\ \texttt{(p} \Vdash \varphi\ \texttt{([x] @ env))}).$$

Equivalent statements, along with the ones corresponding to `Forces_Equal` and `Forces_Nand`, appear in Kunen as the inductive definition of the forcing relation [9, Def. IV.2.42].

As with the previous section, the proofs of the forcing theorems have two different flavors: The ones for the atomic formulas proceed by using the principle of `forces_induction`, and then an induction on `formula` wraps the former with the remaining cases (`Nand` and `Forall`).

As an example of the first class, we can take a look at our formalization of [9, Lem. IV.2.40(a)]. Note that the context (a "locale," in Isabelle terminology, namely `forcing_data`) of the lemma includes the assumption of `P` being a forcing notion, and the predicate of being M-generic is defined in terms of `P`:

lemma *IV240a:*
  assumes
    "M_generic(G)"
  shows
  "(τ∈M⟶ϑ∈M⟶(∀p∈G.forces_eq(p,τ,ϑ)⟶val(G,τ)=val(G,ϑ)))
  ∧
  (τ∈M⟶ϑ∈M⟶(∀p∈G.forces_mem(p,τ,ϑ)⟶val(G,τ)∈val(G,ϑ)))"

Its proof starts by an introduction of `forces_induction`; the inductive cases for each atomic type were handled before as separate lemmas (`IV240a_mem` and `IV240a_eq`). We illustrate with the statement of the latter.

lemma *IV240a_eq:*
assumes
"M_generic(G)" "p∈G" "forces_eq(p,τ,ϑ)"
and
IH:"⋀q σ. q∈P ⟹ q∈G ⟹ σ∈domain(τ) ∪ domain(ϑ) ⟹
(forces_mem(q,σ,τ) ⟶ val(G,σ) ∈ val(G,τ)) ∧
(forces_mem(q,σ,ϑ) ⟶ val(G,σ) ∈ val(G,ϑ))"
shows
"val(G,τ) = val(G,ϑ)"

Examples of proofs using the second kind of induction include the basic `strengthening_lemma` and the main results in this section, the lemmas of Density (actually, its nontrivial direction `dense_below_imp_forces`) and Truth, which we state next.

lemma *density_lemma:*
assumes
"p∈P" "φ∈formula" "env∈list(M)" "arity(φ)≤length(env)"
shows
"(p ⊩ φ env) ⟷ dense_below({q∈P. (q ⊩ φ env)},p)"

lemma *truth_lemma:*
assumes
"φ∈formula" "M_generic(G)"
shows
"⋀env. env∈list(M) ⟹ arity(φ)≤length(env) ⟹
(∃p∈G. (p ⊩ φ env)) ⟷ M[G], map(val(G),env) ⊨ φ"

From these results, the semantical characterization of the forcing relation (the "definition of ⊩" in [9, IV.2.22]) follows easily:

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)"

The present statement of the Fundamental Theorems is almost exactly the same of those in our previous report [4], with the only modification being the bound on arities and a missing typing constraint. This implied only minor adjustments in the proofs of the satisfaction of axioms.

6 Example of Proper Extension

Even when the axioms of *ZFC* are proved in the generic extension, one cannot claim that the magic of forcing has taken place unless one is able to provide some

proper extension with the *same ordinals*. After all, one is assuming from the start a model M of *ZFC*, and in some trivial cases $M[G]$ might end up to be exactly M; this is where *proper* enters the stage. But, for instance, in the presence of large cardinals, a model $M' \supsetneq M$ might be an end-extension of M—this is where we ask the two models to have the same ordinals, the same *height*.

Three theory files contain the relevant results. `Ordinals_In_MG.thy` shows, using the closure of M under ranks, that M and $M[G]$ share the same ordinals (actually, ranks of elements of $M[G]$ are bounded by the ranks of their names in M):

lemma *rank_val: "rank(val(G,x)) ≤ rank(x)"*
lemma *Ord_MG_iff:*
  assumes *"Ord(α)"*
  shows *"α ∈ M ⟷ α ∈ M[G]"*

To prove these results, we found it useful to formalize induction over the relation $\mathtt{ed}(x, y) := x \in \mathrm{domain}(y)$, which is key to arguments involving names.

`Succession_Poset.thy` contains our first example of a poset that interprets the locale `forcing_notion`, essentially the notion for adding one Cohen real. It is the set $2^{<\omega}$ of all finite binary sequences partially ordered by reverse inclusion. The sufficient condition for a proper extension is that the forcing poset is *separative*: every element has two incompatible ($\perp$`s`) extensions. Here, `seq_upd(f,x)` adds `x` to the end of the sequence `f`.

lemma *seqspace_separative:*
  assumes *"f∈2^<ω"*
  shows *"seq_upd(f,0) ⊥s seq_upd(f,1)"*

We prove in the theory file `Proper_Extension.thy` that, in general, every separative forcing notion gives rise to a proper extension.

7 The Axioms of Replacement and Choice

In our report [4] we proved that any generic extension preserves the satisfaction of almost all the axioms, including the separation scheme (we adapted those proofs to the current statement of the axiom schemes). Our proofs that Replacement and choice hold in every generic extension depend on further relativized concepts and closure properties.

7.1 Replacement

The proof of the Replacement Axiom scheme in $M[G]$ in Kunen uses the Reflection Principle relativized to M. We took an alternative pathway, following Neeman [11]. In his course notes, he uses the relativization of the cumulative hierarchy of sets.

The family of all sets of rank less than α is called `Vset`(α) in Isabelle/ZF. We showed, in the theory file `Relative_Univ.thy` the following relativization

and closure results concerning this function, for a class M satisfying the locale `M_eclose` plus the Powerset Axiom and four instances of replacement.

lemma *Vset_abs:* "⟦ *M(i); M(V); Ord(i)* ⟧ ⟹
is_Vset(M,i,V) ⟷ *V = {x∈Vset(i). M(x)}*"

lemma *Vset_closed:* "⟦ *M(i); Ord(i)* ⟧ ⟹ *M({x∈Vset(i). M(x)})*"

We also have the basic result

lemma *M_into_Vset:*
assumes *"M(a)"*
shows *"∃i[M]. ∃V[M]. ordinal(M,i) ∧ is_Vfrom(M,0,i,V) ∧ a∈V"*

stating that M is included in $\bigcup\{\texttt{Vset}^M(\alpha) : \alpha \in M\}$ (actually they are equal).

For the proof of the Replacement Axiom, we assume that φ is functional in its first two variables when interpreted in $M[G]$ and the first ranges over the domain $\texttt{c} \in M[G]$. Then we show that the collection of all values of the second variable, when the first ranges over `c`, belongs to $M[G]$:

lemma *Replace_sats_in_MG:*
assumes
"c∈M[G]" "env ∈ list(M[G])"
"φ ∈ formula" "arity(φ) ≤ 2 #+ length(env)"
"univalent(##M[G], c, λx v. (M[G], [x,v]@env ⊨ φ))"
shows
"{v. x∈c, v∈M[G] ∧ (M[G], [x,v]@env ⊨ φ)} ∈ M[G]"

From this, the satisfaction of the Replacement Axiom in $M[G]$ follows very easily.

The proof of the previous lemma, following Neeman, proceeds as usual by turning an argument concerning elements of $M[G]$ to one involving names lying in M, and connecting both worlds by using the forcing theorems. In the case at hand, by functionality of φ we know that for every $x \in c \cap M[G]$ there exists exactly one $v \in M[G]$ such that $M[G], [x, v]$ @ $env \models \varphi$. Now, given a name $\pi' \in M$ for c, every name of an element of c belongs to $\pi := \text{domain}(\pi') \times \mathbb{P}$, which is easily seen to be in M. We will use π to be the domain in an application of the Replacement Axiom in M. But now, obviously, we have lost functionality since there are many names $\dot{v} \in M$ for a fixed v in $M[G]$. To solve this issue, for each $\rho p := \langle \rho, p \rangle \in \pi$ we calculate the minimum rank of some $\tau \in M$ such that $p \Vdash \varphi(\rho, \tau, \dots)$ if there is one, or 0 otherwise. By Replacement in M, we can show that the supremum `?sup` of these ordinals belongs to M and we can construct a `?bigname` := `{x∈Vset(?sup). x ∈ M} × {one}` whose interpretation by (any generic) G will include all possible elements as v above.

The previous calculation required some absoluteness and closure results regarding the minimum ordinal binder, `Least`(Q), also denoted $\mu x.Q(x)$, that can be found in the theory file `Least.thy`.

7.2 Choice

A first important observation is that the proof of AC in $M[G]$ only requires the assumption that M satisfies (a finite fragment of) ZFC. There is no need to invoke Choice in the metatheory.

Although our previous version of the development used AC, that was only needed to show the Rasiowa-Sikorski Lemma (RSL) for arbitrary posets. We have modularized the proof of the latter and now the version for countable posets that we use to show the existence of generic filters does not require Choice (as it is well known). We also bundled the full RSL along with our implementation of the principle of dependent choices in an independent branch of the dependency graph, which is the only place where the theory `ZF.AC` is invoked.

Our statement of the Axiom of Choice is the one preferred for arguments involving transitive classes satisfying ZF:

$$\forall \texttt{x[M]}.\ \exists \texttt{a[M]}.\ \exists \texttt{f[M]}.\ \texttt{ordinal(M,a)} \wedge \texttt{surjection(M,a,x,f)}$$

The Simplifier is able to show automatically that this statement is equivalent to the next one, in which the real notions of ordinal and surjection appear:

$$\forall \texttt{x[M]}.\ \exists \texttt{a[M]}.\ \exists \texttt{f[M]}.\ \texttt{Ord(a)} \wedge \texttt{f} \in \texttt{surj(a,x)}$$

As with the forcing axioms, the proof of AC in $M[G]$ follows the pattern of Kunen [9, IV.2.27] and is rather straightforward; the only complicated technical point being to show that the relevant name belongs to M. We assume that $\texttt{a} \neq \varnothing$ belongs to $M[G]$ and has a name $\tau \in M$. By AC in M, there is a surjection `s` from an ordinal $\alpha \in M$ ($\subseteq M[G]$) onto domain(τ). Now

$$\texttt{\{opair\_name(check(}\beta\texttt{),s`}\beta\texttt{).}\ \beta \in \alpha\texttt{\}} \times \texttt{\{one\}}$$

is a name for a function `f` with domain α such that `a` is included in its range, and where `opair_name`(σ, ρ) is a name for the ordered pair $\langle val(G,\sigma), val(G,\rho)\rangle$. From this, AC in $M[G]$ follows easily.

7.3 The Main Theorem

With all these elements in place, we are able to transcript the main theorem of our formalization:

```
theorem extensions_of_ctms:
  assumes
    "M ≈ nat" "Transset(M)" "M ⊨ ZF"
  shows
    "∃N.
      M ⊆ N ∧ N ≈ nat ∧ Transset(N) ∧ N ⊨ ZF ∧ M≠N ∧
      (∀α. Ord(α) ⟶ (α ∈ M ⟷ α ∈ N)) ∧
      (M, []⊨ AC ⟶ N ⊨ ZFC)"
```

Here, $\approx$ stands for equipotency, `nat` is the set of natural numbers, and the predicate `Transset` indicates transitivity; and as usual, `AC` denotes the Axiom of Choice, and `ZF` and `ZFC` the corresponding subsets of `formula`.

8 Related Work

There are various formalizations of Zermelo-Fraenkel set theory in proof assistants (v.g. Mizar, Metamath, and recently Lean [10]) that proceed to different levels of sophistication. Isabelle/ZF can be regarded as a notational variant of NGB set theory [9, Sect. II.10], because the schemes of Replacement and Separation feature higher order (free) variables playing the role of formula variables. It cannot be proved that the axioms thus written correspond to first order sentences. For this reason, our relativized versions only apply to set models, where we restrict those variables to predicates that actually come from first order formulas. In that sense, the axioms of the locale `M_ZF` correspond more faithfully to the *ZF* axioms.

Traditional expositions of the method of forcing [7,9] are preceded by a study of relativization and absoluteness. For this reason, it was a natural choice at the beginning of this project to build on top of Paulson's formalization of constructibility on Isabelle/ZF, and that was one of the main early reasons to work on that logic instead of, e.g., HOL—below we discuss other reasons. In any case, our development of forcing does not depend on constructibility itself (in contrast to Cohen's original presentation, in which ground models are initial segments of the constructible hierarchy).

A natural question is whether Isabelle/HOL (with a far more solid framework to work with given its infrastructure and automation) would have been a better choice than Isabelle/ZF. In fact, there are two developments of Zermelo-Fraenkel set theory available on it: `HOLZF` by Obua [12] and `ZFC_in_HOL` by Paulson [16]. But these (logically equivalent) frameworks are higher in consistency strength than Isabelle/ZF. To elaborate on this, both ZF and HOL are axiomatized on top of Isabelle's metalogic *Pure*, which is a version of "intuitionistic higher order logic." In [13] Paulson proves that *Pure* is sound for intuitionistic first order logic, thus it does not add any strength to it. On top of this, the axiomatization of Isabelle/ZF results in a system equiconsistent with *ZFC*. On the other hand, showing the consistency of `HOLZF` (and thus `ZFC_in_HOL`) requires assuming the consistency of *ZFC* plus the existence of an inaccessible cardinal [12, Sect. 3]. We note, in contrast, that our extra running assumption of the existence of a countable transitive model is considerably weaker (directly and consistency-wise) than the existence of an inaccessible cardinal.

Concerning the formalization of the method of forcing, to the best of our knowledge there is only one other that deals with forcing for set theory: the recent *Flypitch* project by Han and van Doorn [5,6], which includes a formalization of the independence of CH using the Lean proof assistant. The Flypitch formalization is largely orthogonal to ours (it is based on Boolean-valued models, which are interpreted into type theory through a variant of the Aczel encoding of set theory), and this precludes a direct comparison of code. But we can highlight some conceptual differences between our development and the corresponding fraction of Flypitch.

A first observation concerns consistency strength. The consistency of Lean requires infinitely many inaccessibles. More precisely, let Lean_n be the theory

of CiC foundations of Lean restricted to n type universes. Carneiro proved in his MSc thesis [1] the consistency of Lean_n from *ZFC* plus the existence of n inaccessible cardinals. It is also reported in Carneiro's thesis that Werner's results in [19] can be adapted to show that Lean_{n+2} proves the consistency of the latter theory. In that sense, although Flypitch includes proofs of unprovability results in first order logic, the meta-theoretic machinery used to obtain them is far heavier than the one we use to operate model-theoretically.

In second place, a formalization of forcing with general partial orders, generic filters and ctms has—in our opinion—the added value that this approach is used in an important (perhaps the greatest) fraction of the literature, both in exposition and in research articles and monographs. In verifying a piece of mature mathematics as the present one, representing the actual practice seems paramount to us.

Finally, as a matter of taste, one of the main benefits of using transitive models is that many fundamental notions are absolute and thus most of the concepts and statements can be interpreted transparently, as we have noted before. It also provides a very concrete way to understand generic objects: as sets that (in the non trivial case) are provably not in the original model; this dispels any mystical feel around this concept (contrary to the case when the ground model is the universe of all sets). In addition, two-valued semantics is closer to our intuition.

9 Conclusion and Future Work

We consider that the formalization of the definition of `forces` and its recursive characterization of forcing for atomic formulas is a turning point in our project; the reason for this is that all further developments will not involve such a daunting metamathematical component. Even the proofs of the Fundamental Theorems of Forcing turned out to follow rather smoothly after this initial setup was ready, the only complicated affair being to show that various dense sets belong to M. Actually, this is a point to be taken care of: For every new concept that is introduced, some lemmas concerning relativization and closure must be proved to be able to synthesize its internal definition. Further automation must be developed for this purpose.

In the course of obtaining internal formulas for the atomic case of forcing, a fruitful discussion concerning complementary perspectives on the role of proof assistants took place. An earlier approach relied more heavily in formula synthesis, thus making the Simplifier an indispensable main character. Following this line was quickier from the coding point of view since few new primitives were introduced and thus fewer lemmas concerning absoluteness and arities. On the downside, processing was a bit slower, the formulas synthesized were gigantic and the process on a whole was more error-prone. In fact, this approach was unsuccessful and we opted for a more detailed engineering, defining all intermediate steps. So the load on the assistant, in this part of the development, balanced from code-production to code-verification.

The next task in our path is pretty clear: To develop the forcing notions to obtain the independence of *CH* along with the prerequisite combinatorial results, v.g. the Δ-system lemma. A development of cofinality is under way in a joint work with E. Pacheco Rodríguez, which is needed for a general statement of the latter. Once these developments are finished, we will be able to give a more thorough comparison between our project and the *Flypitch* approach using Boolean-valued models.

In a second release of `ZF-Constructible-Trans`, we intend to conform it to the lines of *Basic Set Theory (BST)* proposed by Kunen [9, I.3.1] in which elementary results have proofs using alternatively Powerset or Replacement. The interest in this arises because many natural set models (rank-initial segments of the universe or the family $H(\kappa)$ of sets of cardinality less than κ hereditarily) satisfy one of those axioms and not the other. There are also still some older or less significant proofs written in tactical (**apply**) format; we hope we will find the time to translate them to Isar. Finally, the automation of formula synthesis is on an early stage of development; finishing that module will make writing our proofs of closure under various operations faster, and also turn the set theory libraries more usable to other researchers.

References

1. Carneiro, M.: The type theory of Lean. Master's thesis, Carnegie Mellon University, April 2019. https://github.com/digama0/lean-type-theory/releases/tag/v1.0
2. Gödel, K.: The Consistency of the Continuum Hypothesis. Annals of Mathematics Studies, no. 3. Princeton University Press, Princeton (1940)
3. Gunther, E., Pagano, M., Sánchez Terraf, P.: First steps towards a formalization of forcing. In: Accattoli, B., Olarte, C. (eds.) Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2018, Fortaleza, Brazil, 26–28 September 2018. Electronic Notes in Theoretical Computer Science, vol. 344, pp. 119–136. Elsevier (2018). https://doi.org/10.1016/j.entcs.2019.07.008
4. Gunther, E., Pagano, M., Sánchez Terraf, P.: Mechanization of separation in generic extensions. arXiv e-prints arXiv:1901.03313, January 2019
5. Han, J.M., van Doorn, F.: A formalization of forcing and the unprovability of the continuum hypothesis. In: Harrison, J., O'Leary, J., Tolmach, A. (eds.) 10th International Conference on Interactive Theorem Proving (ITP 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 141, pp. 19:1–19:19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2019). https://doi.org/10.4230/LIPIcs.ITP.2019.19
6. Han, J.M., van Doorn, F.: A formal proof of the independence of the continuum hypothesis. In: Blanchette, J., Hritcu, C. (eds.) Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, 20–21 January 2020. ACM (2020). https://doi.org/10.1145/3372885
7. Jech, T.: Set Theory. The Millennium Edition. Springer Monographs in Mathematics, 3rd edn. Springer, Heidelberg (2002). Corrected fourth printing (2006)
8. Kunen, K.: Set Theory: An Introduction to Independence Proofs. Studies in Logic and the Foundations of Mathematics. Elsevier Science, Amsterdam (1980)

9. Kunen, K.: Set Theory. Studies in Logic, 2nd edn. College Publications, London (2011). Revised edition (2013)
10. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The Lean theorem prover (system description). In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 378–388. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_26
11. Neeman, I.: Topics in set theory, forcing (2011). Course Lecture Notes https://bit.ly/2ErcbmI
12. Obua, S.: Partizan games in Isabelle/HOLZF. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) ICTAC 2006. LNCS, vol. 4281, pp. 272–286. Springer, Heidelberg (2006). https://doi.org/10.1007/11921240_19
13. Paulson, L.C.: The foundation of a generic theorem prover. J. Autom. Reason. **5**(3), 363–397 (1989). https://doi.org/10.1007/BF00248324
14. Paulson, L.C. (ed.): Isabelle. LNCS, vol. 828. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0030541
15. Paulson, L.C.: The relative consistency of the axiom of choice mechanized using Isabelle/ZF. LMS J. Comput. Math. **6**, 198–248 (2003). https://doi.org/10.1112/S1461157000000449
16. Paulson, L.C.: Zermelo Fraenkel set theory in higher-order logic. Archive of Formal Proofs, October 2019. http://isa-afp.org/entries/ZFC_in_HOL.html. Formal proof development
17. Paulson, L.C., Grabczewski, K.: Mechanizing set theory. J. Autom. Reason. **17**(3), 291–323 (1996). https://doi.org/10.1007/BF00283132
18. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_7
19. Werner, B.: Sets in types, types in sets. In: Abadi, M., Ito, T. (eds.) TACS 1997. LNCS, vol. 1281, pp. 530–546. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0014566