First steps towards a formalization of Forcing

Emmanuel Gunther[1]

FaMAF
Universidad Nacional de Córdoba
Córdoba, Argentina

Miguel Pagano[2]

FaMAF
Universidad Nacional de Córdoba
Córdoba, Argentina

Pedro Sánchez Terraf[3]

CIEM-FaMAF
Universidad Nacional de Córdoba
Córdoba, Argentina

Abstract

We lay the ground for an Isabelle/ZF formalization of Cohen's technique of *forcing*. We formalize the definition of forcing notions as preorders with top, dense subsets, and generic filters. We formalize a version of the principle of Dependent Choices and using it we prove the Rasiowa-Sikorski lemma on the existence of generic filters.
Given a transitive set M, we define its generic extension $M[G]$, the canonical names for elements of M, and finally show that if M satisfies the axiom of pairing, then $M[G]$ also does. We also prove $M[G]$ is transitive.

Keywords: Isabelle/ZF, forcing, preorder, Rasiowa-Sikorski lemma, names, generic extension.

1 Introduction

Set Theory plays a double role in Mathematics: It is one of its possible foundations and also an active research area. As it is widely known, Georg Cantor introduced its main concepts and in particular showed the fundamental result that the real line, $\mathbb{R}$ is not equipotent to the natural numbers. Soon after this, he posed the most important question in the field, written as a conjecture:

The *Continuum Hypothesis* (*CH*). Every uncountable subset of $\mathbb{R}$ is equipotent to $\mathbb{R}$.

The current axiomatic foundation of Set Theory is through first-order logic and uses the axioms devised by Zermelo and Fraenkel, including the Axiom of Choice (*AC*) among them. This theory is known by the *ZFC* acronym. Gödel [3] showed that *CH* cannot be refuted using *ZFC*, unless this theory itself is inconsistent (we say that *CH is relatively consistent with ZFC*). For a while, this result left the possibility that one might be able to show $ZFC \models CH$, but in a groundbreaking work [2], Paul Cohen discovered the technique of *forcing* and

[1] Email: gunther@famaf.unc.edu.ar
[2] Email: pagano@famaf.unc.edu.ar
[3] Email: sterraf@famaf.unc.edu.ar
[4] Supported by Secyt-UNC project 33620180100465CB.

proved that $\neg CH$ is relatively consistent with ZFC. Forcing has been used since then for showing innumerable independence results and to perform mathematical constructions.

A great part of Gödel's work on this subject has been formalized in Isabelle [19] by Lawrence Paulson [10]. This paper formalizes a first part of the machinery of forcing, mostly by following the new edition of the classical book on the subject by Kunen [6]. In the rest of the introduction we discuss some of the set-theoretical details involved and explain briefly Paulson's formalization.

1.1 Models of ZFC

By Gödel's Second Incompleteness Theorem, we cannot prove that there exists a model of ZFC. More formally, if we assume that mathematical proofs can be encoded as theorems of ZFC and that the latter do not lead to contradictions (i.e., ZFC is *consistent*), then we cannot prove that there exists a set M and a binary relation E such that $\langle M, E\rangle$ satisfies the ZFC axioms.

A relative consistency proof for an axiom A is then obtained by assuming that there exists a model of ZFC, say $\langle M, E\rangle$, and constructing another model $\langle M', E'\rangle$ for $ZFC + A$. We single out a very special kind of models:

Definition 1.1 (i) A set M (of sets) is *transitive* if for all $x \in M$ and $y \in x$, we have $y \in M$ (i.e., every element of M is a subset of M).

(ii) $\langle M, E\rangle$ is a *transitive model* if M is transitive and E is the membership relation $\in$ restricted to M. It is *countable* if M is equipotent to a subset of $\mathbb{N}$; we then say that the model M is a *ctm*.

As in the last sentence, one usually refers to a transitive model by the underlying set because the relation is fixed.

In spite of Gödel's Second Incompleteness Theorem, one can find transitive models for every finite fragment of ZFC. More precisely,

Theorem 1.2 *For each finite subset $\Phi \subseteq ZFC$, the statement* "there exists a countable transitive model of Φ" *is a theorem of ZFC.*

This follows by a combination of the Reflection Principle, the Löwenheim-Skolem Theorem, and the Mostowksi Collapse. The reader can consult the details in [6]. Consistency arguments that assume the existence of a ctm M of ZFC can usually be replaced by a model as in Theorem 1.2, since a first-order proof (e.g. of a contradiction) [5] involves only finitely many axioms.

It is instructive to sketch Gödel's argument of the relative consistency of CH: Assuming that M is a ctm of ZFC, Gödel showed that M contains a minimal submodel L^M of the same "height" (i.e. having the same ordinals) that satisfies $ZFC + CH$. The sets in L^M are called *constructible* and are in a sense "definable." In fact, there is a first-order formula L such that $L^M = \{x \in M : M \models L(x)\}$. To show that $L^M \models ZFC + CH$, one uses the fact that ZFC holds in M.

It is therefore a primary need to have means to correlate (first-order) properties satisfied by a model M and those of a submodel $N \subseteq M$. As a simple example on this, consider $M := \{a, b, c, \{a, b\}, \{a, b, c\}\}$ and $N := \{a, b, \{a, b, c\}\}$, and let

$$\varphi(x, y, z) := \forall w.\, (w \in z \longleftrightarrow w = x \vee w = y).$$

Then we have

$$M \not\models \varphi(a, b, \{a, b, c\}) \quad \text{but} \quad N \models \varphi(a, b, \{a, b, c\}).$$

There is a discrepancy between M and N about $\{a, b, c\}$ being "the (unordered) pair of a and b." We say that φ holds for $a, b, \{a, b, c\}$ *relative* to N. It is immediate to see that φ holds for x, y, z relative to N if and only if

$$\varphi^N(x, y, z) := \forall w.\; w \in N \longrightarrow (w \in z \longleftrightarrow w = x \vee w = y)$$

holds. φ^N is called the *relativization of φ to N*. One can generalize this operation of relativization to the class of all sets satisfying a first-order predicate C in a straightforward way:

$$\varphi^C(x, y, z) := \forall w.\; C(w) \longrightarrow (w \in z \longleftrightarrow w = x \vee w = y)$$

[5] It is relevant to this point that both the approaches by Gödel and Cohen for showing relative consistency of an axiom A can be used to obtain an algorithm transforming a proof concluding a contradiction from $ZFC + A$ to one from ZFC.

It can be shown elementarily that if M and N are transitive, φ^N holds if and only if φ^M holds, for $x, y, z \in N$. We say then that φ is *absolute between N and M*. The concepts of relativization and absoluteness are central to the task of transferring truth of axioms in M to L^M, and constitute the hardest part of Paulson's development.

1.2 Forcing

Forcing is a technique to extend countable transitive models of *ZFC*. This process is guaranteed to preserve the *ZFC* axioms while allowing to fine-tune what other first-order properties the extension will have. Given a ctm M of *ZFC* and a set G, one constructs a new ctm $M[G]$ that includes M and contains G, and proves that under some hypotheses (G being "generic"), $M[G]$ satisfies *ZFC*.

The easiest way to define genericity is by using a preorder with top $\langle \mathbb{P}, \leq, \mathbb{1} \rangle$ in M. In Section 3 we formalize the definitions of *dense* subset and *filter* of $\mathbb{P}$, and we say that G is an M-generic filter if it intersects every dense subset of $\mathbb{P}$ that lies in M.

The Rasiowa-Sikorski lemma (RSL) states that for any preorder $\mathbb{P}$ and any countable family $\{\mathcal{D}_n : n \in \mathbb{N}\}$ of dense subsets of $\mathbb{P}$ there is a filter intersecting every $\mathcal{D}_i$. Thus, there are generic filters G for countable transitive models. In general, no such G belongs to M and therefore the extension $M[G]$ is proper. We formalize the proof of RSL in Section 3.2. A requisite result on a version of the Axiom of Choice is formalized in Section 3.1. We then apply RSL to prove the existence of generic filters in Section 4.1.

Every $y \in M[G]$ is obtained from an element $\dot{y}$ of M, thus elements of M are construed as *names* or codes for elements of $M[G]$. The decoding is given by the function *val*, which takes the generic filter G as a parameter. To prove that M is contained in $M[G]$ it suffices to give a name for each element of M; we define the function *check* which assigns a name for each $x \in M$. Showing that $check(x) \in M$ when $x \in M$ involves some technical issues that will be addressed in a further work. We explain names, *val*, and *check* in Section 4.2.

A central part of this formalization project involves showing that *ZFC* holds in the generic extension. This is most relevant since forcing is essentially the only known way to *extend* models of *ZFC* (while preserving ordinals). The most difficult step to achieve this goal is to define the *forcing relation*, that allows to connect satisfaction in M to that of $M[G]$; this is needed to show that the Powerset axiom and the axiom schemes of Separation and Replacement hold in $M[G]$. In Section 5 we tackle the Pairing Axiom. This does not require the forcing relation, but provides an illustration of the use of names. The development can be downloaded from `https://cs.famaf.unc.edu.ar/~mpagano/forcing/`.

1.3 Related work

Formalization of mathematics serves many purposes [16]. The most obvious one is to increase reliability in a result and/or its proof. This has been the original motivation that lead Voevodsky to gather many researchers around *homotopy type theory* and its formalization in Coq [17]; the same applies to the four color theorem (checked by Gonthier [4]) and the formidable *Flyspeck* project [5] by the team conducted by Hales.

In our particular case, forcing and the set theoretic techniques that are being formalized can be regarded as a mature technology and thus the main goal is not to increase confidence. Nevertheless, the level of detail in a formalization of this sort always provide additional information about the inner workings of the theory: It is expected, for instance, to have a detailed account of which axioms are necessary to define and use forcing. Finally, we support the vision that a growing corpus of formalized mathematics can be a useful library for the future generations. The question of how to systematize this corpus is an ongoing project by Paulson [11].

We will now discuss very succinctly recent formalizations of set theory and forcing. The closest formalizations are those based on Isabelle. Let us remark that Isabelle allows for different logical foundations; in particular, Paulson carried out his formalizations on top of Isabelle/FOL which is based on first-order logic.

There is another major framework in Isabelle based on higher order logic, Isabelle/HOL. This framework is very active, and as a consequence more automated tools are available. Isabelle/HOL has basic chapters on set theory. One of those, by Steven Obua, proceeds up to well founded relations and provides translations between types in HOL (for instance `nat`) to sets (elements of type `ZF`). Another one, by A. Popescu and D. Traytel, reaches cardinal arithmetic. This is fairly limited for our purposes.

Concerning automation, B. Zhan has developed a new tool called `auto2` and applied it to untyped set theory [20]. He has redeveloped essentially the basic results in Isabelle/ZF, but goes in a different direction. Nevertheless, a majority of results in Isabelle/ZF are not yet implemented using this tool, and another downside is that proofs using it do not follow the standard Isar language (see Section 2).

As far as we know, there is little progress on formalizations of forcing in type theory. Most relevant is the work by K. Quirin [14], where a sheaf-theoretic initial approach to forcing is implemented in Coq. This language is extremely different to the standard approach of constructing models of *ZFC*, and it might be difficult (once the forcing machinery is set) to translate results in the literature using ctms to this one. In any case, the translation to set theory of what Quirin accomplishes is to define a generic extension (where *CH*

should fail) and to construct a set K (a candidate counterexample) and injections $\mathbb{N} \hookrightarrow K$ and $K \hookrightarrow \mathbb{R}$. But the most important part, that is, that there are no surjections $\mathbb{N} \twoheadrightarrow K$ and $K \twoheadrightarrow \mathbb{R}$, is left for a future work.

2 Isabelle/ZF

Let us introduce briefly Paulson's formalization of ZF [12] in Isabelle and the main aspects of his formal proof for the relative consistency of the Axiom of Choice [10]; we will only focus on those aspects that are essential to keep this paper self-contained, and refer the interested reader to Paulson's articles. Isabelle/ZF includes a development of classical first-order logic, FOL. Both of them are built upon the core library *Pure*.

In Isabelle/ZF sets are *individuals*, i.e. terms of type `i` and formulas have type `o` (akin to a *Bool* type, but at the object level). The axiomatization of *ZFC* in Isabelle/ZF proceeds by postulating a binary predicate $\in$ and several set constructors (terms and functions with values in `i`) corresponding to the empty set (the constant `0`), powersets, and one further constant `inf` for an infinite set. The axioms, being formulas, are terms of type `o`; the foundation axiom, for example, is formalized as (the universal closure of) `"A = 0 ∨ (∃x∈A. ∀y∈x. y∉A)"`. Besides the axioms, Isabelle/ZF also introduces several definitions (for example, pairs and sets defined by comprehension using separation) and syntactic abbreviations to keep the formalization close to the customary manner of doing mathematics. Working with the library and extending it is quite straightforward. As an example, we introduce a new term-former (which is a combination of instances of replacement and separation) denoting the image of a function over a set defined by comprehension, namely $\{b(x) : x \in A \text{ and } Q(x)\}$:

definition *SepReplace :: "[i, i⇒i, i⇒ o] ⇒i"* **where**
"SepReplace(A,b,Q) == {y . x∈A, y=b(x) ∧ Q(x)}"

We are then able to add the abbreviation *{b .. x∈A, Q}* as a notation for *SepReplace(A,b,Q)*. The characterization of our new constructor is given by

lemma *Sep_and_Replace: "{b(x) .. x∈A, Q(x) } = {b(x) . x∈{y∈A. Q(y)}}"*

We now discuss relativization in Isabelle/ZF. Relativized versions of the axioms can be found in the formalization of constructibility [10]. For example, the relativized Axiom of Foundation is

definition *foundation_ax :: "(i=>o) => o"* **where**
"foundation_ax(M) ==
∀x[M]. (∃y[M]. y∈x) ⟶ (∃y[M]. y∈x & ~(∃z[M]. z∈x & z ∈ y))"

The relativized quantifier *∀x[M]. P(x)* is a shorthand for *∀x. M(x) ⟶ P(x)*. In order to express that a (set) model satisfies this axiom we use the "coercion" *## :: i => (i => o)* (that maps a set A to the predicate $\lambda x.(x \in A)$) provided by Isabelle/ZF. As a trivial example we can show that the empty set satisfies Foundation:

lemma *emp_foundation : "foundation_ax(##0)"*

Mathematical texts usually start by fixing a context that defines parameters and assumptions needed to develop theorems and results. In Isabelle the way of defining contexts is through *locales* [1]. Locales can be combined and extended by adding more parameters and assuming more facts, leading to a new locale. For example a context describing lattices can be extended to distributive lattices. The way to instantiate a locale is by *interpreting* it, which consists of giving concrete values to parameters and proving the assumptions. In our work, we use locales to organize the formalization and to make explicit the assumptions of the most important results.

Let us close this section with a brief comment about the facilities provided by the Isabelle framework. The edition is done in an IDE called `jEdit`, which is bundled with the standard Isabelle distribution; it offers the user a fair amount of tools in order to manage theory files, searching for theorems and concepts spread through the source files, and includes tracing utilities for the automatic tools. A main feature is a window showing the *proof state*, where the active (sub)goals are shown, along with the already obtained results and possibly errors.

Isabelle proofs can be written in two dialects. The older one, and also more basic, follows a procedural approach, where one applies several tactics in order to decompose the goal into simpler ones and then solving them (with the aid of automation); the original work by Paulson used this method. Under this approach proofs are constructed top-down resulting in proof-scripts that conceal the mathematical reasoning behind the proof, since the intermediate steps are only shown in the proof state. For this reason, the proof language *Isar* was developed, starting with Wenzel's work [18]. Isar is mostly declarative, and its main purpose is to construct *proof documents* that (in principle) can be read and understood without the need of running the code.

We started this development using the procedural approach, but soon after we realized that for our purposes the Isar language was far more appropriate.

3 Forcing notions

In this section we present a proof of the Rasiowa-Sikorski lemma which uses the principle of dependent choices. We start by introducing the necessary definitions about preorders; then, we explain and prove the principle of dependent choice most suitable for our purpose.

It is to be noted that the order of presentation of the material deviates a bit from the dependency of the source files. The file containing the most basic results and definitions that follow imports that containing the results of Subsection 3.1.

Definition 3.1 A preorder on a set P is a binary relation $\leqslant$ which is reflexive and transitive.

The preorder relation will be represented as a set of pairs, and hence it is a term of type `i`.

Definition 3.2 Given a preorder $(P, \leqslant)$ we say that two elements p, q are *compatible* if they have a lower bound in P. Notice that the elements of P are also sets, therefore they have type `i`.

```
definition compat_in :: "i⇒i⇒i⇒i⇒o" where
  "compat_in(P,leq,p,q) == ∃d∈P . ⟨d,p⟩∈leq ∧ ⟨d,q⟩∈leq"
```

Definition 3.3 A *forcing notion* is a preorder $(P, \leqslant)$ with a maximal element $\mathbb{1} \in P$.

```
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"
```

The locale `forcing_notion` introduces a mathematical context where we work assuming the forcing notion $(P, \leqslant, \mathbb{1})$. In the following definitions we are in the locale `forcing_notion`.

A set D is *dense* if every element $p \in P$ has a lower bound in D and there is also a weaker definition which asks for a lower bound in D only for the elements below some fixed element q.

```
definition dense :: "i⇒o" where
  "dense(D) == ∀p∈P. ∃d∈D . ⟨d,p⟩∈leq"

definition dense_below :: "i⇒i⇒o" where
  "dense_below(D,q) == ∀p∈P. ⟨p,q⟩∈leq ⟶ (∃d∈D . ⟨d,p⟩∈leq)"
```

Since the relation $\leqslant$ is reflexive, it is obvious that P is dense. Actually, this follows automatically once the appropriate definitions are unfolded:

```
lemma P_dense: "dense(P)"
  using leq_preord
  unfolding preorder_on_def refl_def dense_def
  by blast
```

Here, the automatic tactic `blast` solves the goal. In the procedural approach, goals are refined with the command **apply** *tactic*, and proofs are finished using **done**. Then **by** ... is an idiom for **apply** ... **done**.

We say that $F \subseteq P$ is increasing (or upward closed) if every extension of any element in F is also in F.

```
definition increasing :: "i⇒o" where
  "increasing(F) == ∀x∈F. ∀ p∈P . ⟨x,p⟩∈leq ⟶ p∈F"
```

A filter is an increasing set G with all its elements being compatible in G.

```
definition filter :: "i⇒o" where
  "filter(G) == G⊆P ∧ increasing(G) ∧ (∀p∈G. ∀q∈G. compat_in(G,leq,p,q))"
```

We finally introduce the upward closure of a set and prove that the closure of A is a filter if its elements are compatible in A.

```
definition upclosure :: "i⇒i" where
  "upclosure(A) == {p∈P. ∃a∈A. ⟨a,p⟩∈leq}"
lemma closure_compat_filter:  "A⊆P ⟹ (∀p∈A. ∀q∈A. compat_in(A,leq,p,q)) ⟹
filter(upclosure(A))"
```

As usual with procedural proofs, the refinement process goes "backwards," from the main goal to simpler ones. The proof of this last lemma takes 21 lines and 34 proof commands and is one of the longest procedural proofs

in the development. It was at the moment of its implementation that we realized that a declarative approach was best because, apart from being more readable, the reasoning flows mostly in a forward fashion.

3.1 A sequence version of Dependent Choices

The Rasiowa-Sikorski lemma follows naturally from a "pointed" version of the *Principle of Dependent Choices (DC)* which, in turn, is a consequence of the Axiom of Choice (AC). It is therefore natural to take as a starting point the theory `AC` which adds the latter axiom to the toolkit of Isabelle/ZF.

The statement we are interested in is the following:

(Pointed DC) Let R be a binary relation on A, and $a \in A$. If $\forall x \in A.\, \exists y \in A.\, x\ R\ y$, then there exists $f : \omega \to A$ such that $f(0) = a$ and $f(n)\ R\ f(n+1)$ for all $n \in \omega$.

Two different versions of DC (called $DC0$ and $DC(\kappa)$) have already been formalized by Krzysztof Grabczewski [13], as part of a study of equivalents of AC (following Rubin and Rubin [15]). Nevertheless, those are not convenient for our purposes. In fact, the axiom $DC0$ corresponds essentially to our Pointed DC but without the constraint $f(0) = a$; it is a nice exercise to show that $DC0$ implies Pointed DC, but a formalization would have a moderate length. On the other hand, $DC(\kappa)$ is rather different in nature and it is tailored to obtain another proposition equivalent to the axiom of choice (actually, $AC \longleftrightarrow (\forall \kappa.\, \mathrm{Card}(\kappa) \longrightarrow DC(\kappa))$). Finally, the shortest path from AC to $DC0$ using already formalized material involves a complicated detour (130+ proof commands spanning various files of the `ZF-AC` theory and going through the Well Ordering Theorem and $DC(\omega)$), compared to the mere 11 commands from AC to `AC_func_Pow`. This last one is the choice principle that we use in our formalization of Pointed DC, and states the existence of choice functions ("selectors") on $\mathcal{P}(A) \setminus \{\varnothing\}$):

$$\exists (s : \mathcal{P}(A) \setminus \{\varnothing\} \to A).\, \forall X \subseteq A.\ X \neq \varnothing \longrightarrow s(X) \in X.$$

Another advantage of taking `AC_func_Pow` as a starting point is that it does not involve proper classes: The version of AC in Isabelle/ZF corresponds to an axiom scheme of first-order logic and as such is not a standard formulation.

The strategy to prove Pointed DC (following a proof in Moschovakis [7]) is to define the function f discussed above by primitive recursion on the naturals, which can be done easily thanks to the package of Isabelle/ZF [8,9] for definitions by recursion on inductively defined sets. [6]

```
consts dc_witness :: "i ⇒ i ⇒ i ⇒ i ⇒ i ⇒ i"
primrec
  wit0   : "dc_witness(0,A,a,s,R) = a"
  witrec : "dc_witness(succ(n),A,a,s,R) = s`{x∈A. ⟨dc_witness(n,A,a,s,R),x⟩∈R }"
```

Besides the natural argument and the parameters A, a, and R, the function `dc_witness` has a function s as a parameter. If this function is a selector for $\mathcal{P}(A) \setminus \{\varnothing\}$, the function $f(n) := \texttt{dc\_witness}(n, A, a, s, R)$ will satify DC. Notice that s is a term of type `i` (a function construed as a set of pairs) and an expression `s`b` is notation for `apply(s,b)`, where `apply :: "i ⇒ i ⇒ i"` is the operation of function application.

The proof is mostly routine; after a few lemmas (26 proof commands in total) we obtain the following theorem:

```
theorem pointed_DC  : "(∀x∈A. ∃y∈A. ⟨x,y⟩∈ R) ⟹
                       ∀a∈A. (∃f ∈ nat→A. f`0 = a ∧ (∀n ∈ nat. ⟨f`n,f`succ(n)⟩∈R))"
```

We need a further, "diagonal" version of DC to prove Rasiowa-Sikorski. That is, if the assumption holds for a sequence of relations S_n, then $f(n)\ S_{n+1}\ f(n+1)$ for all n.

We first obtain a corollary of DC changing A for $A \times \texttt{nat}$, whose procedural proof takes 16 lines:

```
corollary DC_on_A_x_nat :
  "(∀x∈A×nat. ∃y∈A. ⟨x,⟨y,succ(snd(x))⟩⟩ ∈ R) ⟹
    ∀a∈A. (∃f ∈ nat→A. f`0 = a ∧ (∀n ∈ nat. ⟨⟨f`n,n⟩,⟨f`succ(n),succ(n)⟩⟩∈R))"
```

The following lemma is then proved automatically:

```
lemma aux_sequence_DC : "∀x∈A. ∀n∈nat. ∃y∈A. ⟨x,y⟩ ∈ S`n ⟹
  ∀x∈A×nat. ∃y∈A. ⟨x,⟨y,succ(snd(x))⟩⟩ ∈ {⟨⟨w,n⟩,⟨y,m⟩⟩∈(A×nat)×(A×nat). ⟨w,y⟩∈S`m }"
  by auto
```

[6] The package figures out the inductive set at hand and checks that the recursive definition makes sense; for example, it rejects definitions with a missing case.

And after a short proof we arrive to DC for a sequence of relations:

```
lemma sequence_DC: "∀x∈A. ∀n∈nat. ∃y∈A. ⟨x,y⟩ ∈ S‘n ⟹
    ∀a∈A. (∃f ∈ nat→A. f‘0 = a ∧ (∀n ∈ nat. ⟨f‘n,f‘succ(n)⟩∈S‘succ(n)))"
  apply (drule aux_sequence_DC)
  apply (drule DC_on_A_x_nat, auto)
  done
```

3.2 The Rasiowa-Sikorski lemma

In order to state this Lemma, we gather the relevant hypotheses into a locale:

```
locale countable_generic = forcing_notion +
  fixes 𝒟
  assumes countable_subs_of_P:  "𝒟 ∈ nat→Pow(P)"
  and     seq_of_denses:        "∀n ∈ nat. dense(𝒟‘n)"
```

That is, $\mathcal{D}$ is a sequence of dense subsets of the poset P. A filter is $\mathcal{D}$-*generic* if it intersects every dense set in the sequence.

```
definition  D_generic :: "i⇒o" where
  "D_generic(G) == filter(G) ∧ (∀n∈nat.(𝒟‘n)∩G≠0)"
```

We can now state the Rasiowa-Sikorski Lemma.

```
theorem rasiowa_sikorski:
  "p∈P ⟹ ∃G. p∈G ∧ D_generic(G)"
```

The intuitive argument for the result is simple: Once $p_0 = p \in P$ is fixed, we can recursively choose p_{n+1} such that $p_n \geq p_{n+1} \in \mathcal{D}_n$, since $\mathcal{D}_n$ is dense in P. Then the filter generated by $\{p_n : n \in \omega\}$ intersects each set in the sequence $\{\mathcal{D}_n\}_n$. This argument appeals to the sequence version of DC; we have to prove first that the relevant relation satisfies its hypothesis:

```
lemma RS_relation:
  assumes
        1:  "x∈P"
            and
        2:  "n∈nat"
  shows
            "∃y∈P. ⟨x,y⟩ ∈ (λm∈nat. {⟨x,y⟩∈P*P. ⟨y,x⟩∈leq ∧ y∈𝒟‘(pred(m))}‘n"
```

These two proofs have been implemented using the Isar proof language.

4 The generic extension

Cohen's technique of forcing consists of constructing new models of ZFC by adding a *generic* subset G of the forcing notion P (a preorder with top). Given a model M of ZFC, the extension with the generic subset G is called *the generic extension* of M, denoted $M[G]$. In this section we introduce all the necessary concepts and results for defining $M[G]$; namely, we show, using Rasiowa-Sikorski, that every preorder in a ctm admits a generic filter and also develop the machinery of names. As an application of the latter, we prove some basic results about the generic extension.

4.1 The generic filter

The following locale gathers the data needed to ensure the existence of an M-generic filter for a poset `P`.

```
locale forcing_data = forcing_notion +
  fixes M enum
  assumes M_countable:      "enum∈bij(nat,M)"
      and P_in_M:           "P ∈ M"
      and leq_in_M:         "leq ∈ M"
      and trans_M:          "Transset(M)"
```

An immediate consequence of the Rasiowa-Sikorski Lemma is the existence of an M-generic filter for a poset `P`.

```
lemma generic_filter_existence:
  "p∈P ⟹ ∃G. p∈G ∧ M_generic(G)"
```

By defining an appropriate countable sequence of dense subsets of `P`,

```
let
        ?D="λn∈nat. (if (enum‘n⊆P ∧ dense(enum‘n))  then enum‘n else P)"
```

we can instantiate the locale `countable_generic`

```
have
      Eq2: "∀n∈nat. ?D‘n ∈ Pow(P)"
  by auto
then have
      Eq3: "?D:nat→Pow(P)"
  by (rule lam_codomain)
have
      Eq4: "∀n∈nat. dense(?D‘n)"
```

...

```
from Eq3 and Eq4 interpret
       cg: countable_generic P leq one ?D
  by (unfold_locales, auto)
```

and then a $\mathcal{D}$-generic filter given by Rasiowa-Sikorski will be M-generic by construction.

```
from cg.rasiowa_sikorski and Eq1 obtain G where
      Eq6: "p∈G ∧ filter(G) ∧ (∀n∈nat.(?D‘n)∩G≠0)"
  unfolding cg.D_generic_def by blast
then have
      Eq7: "(∀D∈M. D⊆P ∧ dense(D) ⟶ D∩G≠0)"
```

We omit the rest of this Isar proof.

4.2 Names

We formalize the function *val* that allows to construct the elements of the generic extension $M[G]$ from elements of the ctm M and the generic filter G. The definition of *val* can be written succinctly as a recursive equation

$$val(G,\tau) := \{val(G,\sigma) : \exists p \in \mathbb{P}.\, (\langle \sigma, p\rangle \in \tau \wedge p \in G)\}. \tag{1}$$

The justification that *val* is well-defined comes from a general result (transfinite recursion on well-founded relations [6, p. 48]). Given a well-founded relation $R \subseteq A \times A$ and a functional $H : A \times (A \to A) \to A$, the principle asserts the existence of a function $F : A \to A$ satisfying $F(a) = H(a, F \restriction (R^{-1}(a)))$. This principle is formalized in Isabelle/ZF and one can use the operator `wfrec` [7] to define functions using transfinite recursion. To be precise, `wfrec :: [i, i, [i,i]=>i] => i` is a slight variation, where the first argument is the relation, the third is the functional, and the second corresponds to the argument of F. Notice that the relation and the function argument of the functional are internalized as terms of type `i`.

In our case the functional is called Hv and takes an additional argument for the parameter G:

$$Hv(G,y,f) = \{f(x) : x \in dom(y) \wedge \exists p \in \mathbb{P}.\, (\langle x, p\rangle \in y \wedge p \in G)\}$$

while the relation is given by:

$$x \; ed \; y \iff \exists p.\langle x, p\rangle \in y.$$

Recall that in *ZFC*, an ordered pair $\langle x, y\rangle$ is the set $\{\{x\}, \{x, y\}\}$. It is trivial to deduce the well-foundedness of *ed* from the fact that $\in$ is well-founded, which follows from the Foundation Axiom.

In our formalization of this recursion, the first argument of `wfrec` is the term of type `i` obtained by restricting the relation *ed* to a set:

```
definition
  edrel :: "i ⇒ i" where
  "edrel(A) == {<x,y> ∈ A*A . x ∈ domain(y)}"
```

Since `edrel(A)` is a subset of a well-founded relation (the transitive closure of the membership relation restricted to `A`), then it is well-founded as well.

[7] Notice that this form of recursive definitions is more general than the one used in the previous section to define `dc_witness`.

```
lemma wf_edrel : "wf(edrel(A))"
  apply (rule wf_subset [of "trancl(Memrel(eclose(A)))"])
  apply (auto simp add:edrel_sub_memrel wf_trancl wf_Memrel)
  done
```

All but one lemma used in the above proof (`wf_subset`, `wf_trancl`, and `wf_Memrel`) are already present in Isabelle/ZF. The remaining technical result has been proved using the Isar language:

```
lemma edrel_sub_memrel: "edrel(A) ⊆ trancl(Memrel(eclose(A)))"
```

The formalization of the functional Hv is straightforward and val is defined using `wfrec`:

```
definition
  Hv :: "i⇒i⇒i⇒i" where
  "Hv(G,y,f) == { f`x .. x∈ domain(y), ∃p∈P. <x,p> ∈ y ∧ p ∈ G }"

definition
  val :: "i⇒i⇒i" where
  "val(G,τ) == wfrec(edrel(eclose(M)), τ, Hv(G))"
```

Then we can recover the recursive expression (1) thanks to the following lemma:

```
lemma def_val:
       "x∈M ⟹ val(G,x) = {val(G,t) .. t∈domain(x) , ∃p∈P .  ⟨t, p⟩∈x ∧ p ∈ G }"
```

We can finally define the generic extension of M by G, also setting up the notation $M[G]$ for it:

```
definition
  GenExt :: "i⇒i"     ("M[_]") where
  "GenExt(G)== {val(G,τ). τ ∈ M}"
```

It is conventional in Isabelle/ZF to define introduction and destruction rules for definitions like `GenExt`; in our case, it is enough to know $x \in M$ in order to know $val(G, x) \in M[G]$:

```
lemma GenExtI:  "x ∈ M ⟹ val(G,x) ∈ M[G]"
```

The destruction rule corresponding to the generic extension says that any $x \in M[G]$ comes from some $\tau \in M$ via val.

```
lemma GenExtD:  "x ∈ M[G] ⟹ ∃τ∈M. x = val(G,τ)"
```

We now provide names for elements in M. That is, for each $x \in M$, we define $check(x)$ (usually denoted by $\check{x}$ in the literature) such that $val(G, check(x)) = x$. This will show that $M \subseteq M[G]$, with a caveat we make explicit in the end of this section. As explained in the introduction, the fact that $M[G]$ extends M is crucial to show that ZFC holds in the former. The definition of $check(x)$ is a straightforward $\in$-recursion:

$$check(x) := \{\langle check(y), \mathbb{1}\rangle : y \in x\} \tag{2}$$

Now the set-relation argument for `wfrec` is the membership relation restricted to a set `A`, `Memrel(A)`.

```
definition
  Hcheck :: "[i,i] ⇒ i" where
  "Hcheck(z,f)  == { <f`y,one> . y ∈ z}"

definition
  check :: "i ⇒ i" where
  "check(x) == wfrec(Memrel(eclose({x})), x , Hcheck)"
```

Here, `eclose` returns the (downward) $\in$-closure of its argument. The main lemmas about val and $check$ require some instances of replacement for M; we set up a locale to assemble these assumptions:

```
locale M_extra_assms = forcing_data +
  assumes check_in_M :       "⋀x. x ∈ M ⟹ check(x) ∈ M"
      and sats_upair_ax:     "upair_ax(##M)"
      and repl_check_pair : "strong_replacement(##M,λp y. y =<check(p),p>)"
```

The first assumption asserts that all the relevant names are indeed in M (i.e., $check(x) \in M$ if $x \in M$) and it is needed to prove that $val(G, check(x)) = x$. It will take a serious effort to fulfill this assumption: One of the hardest parts of Paulson's formalization of constructibility involves showing that models are closed under recursive construction. We will eventually formalize that if $M \models ZFC$ and the arguments of `wfrec` are in M,

then its value also is. This will require to adapt to ctm models several locales defined in [10] that were intended to be used for the class of constructible sets. Notice that the only requirement on the set `G` is that it contains the top element of the poset `P`.

lemma *valcheck :*
  assumes *"one ∈ G"*
  shows *"y ∈ M ⟹ val(G,check(y)) = y"*

4.3 Basic results about the generic extension

We turn now to prove that $M[G]$ is transitive and $G \in M[G]$. Showing that $M[G]$ is transitive amounts to prove $y \in M[G]$ for any $x \in M[G]$ and $y \in x$.

lemma *trans_Gen_Ext' :*
  assumes *"x ∈ M[G]"* **and** *"y ∈ x"*
  shows *"y ∈ M[G]"*

The proof of this lemma is straightforward because from $x \in M[G]$ we can obtain $\tau \in M$ such that $x = val(G, \tau)$. Notice also that using the characterization of *val* given by `def_val` we can extract some $\theta \in \mathrm{dom}(\tau)$ such that $y = val(G, \theta)$; to conclude $val(G, \theta) \in M[G]$ it is enough to prove $\theta \in M$, which follows from the transitivity of M.

In contrast, the proof that $G \in M[G]$ is more demanding. In fact, we set $\dot{G} = \{\langle \check{p}, p\rangle \mid p \in P\}$ as a putative name for G. Proving that $\dot{G}$ is in fact a name for G requires to prove that $\dot{G} \in M$, using an instance of replacement for M (namely that given by the assumption `repl_check_pair`), and then proving that $val(G, \dot{G}) = G$.

definition
  G_dot :: "i" **where**
  "G_dot == {<check(p),p> . p∈P}"

lemma *G_dot_in_M : "G_dot ∈ M"*

lemma *val_G_dot :*
  assumes *"G ⊆ P"* **and** *"one ∈ G"*
  shows *"val(G,G_dot) = G"*

5 Pairing in the generic extension

In this section we show that the generic extension satisfies the pairing axiom; the purpose of this section is to show how to prove that $M[G]$ models one of the axioms of *ZFC*, assuming that M satisfies *ZFC*.[8] In the locale `M_extra_assms` we stated the assumption `sats_upair_ax` which captures that M satisfies pairing. We use *relativized* versions of the axioms in order to express satisfaction.

As we have already mentioned, in Paulson's library, the relativized versions of the *ZFC* axioms are defined for classes (which are defined as predicates over sets). The definition `upair_ax` corresponds to the Pairing Axiom:

definition
  upair :: "[i⇒o,i,i,i] ⇒ o" **where**
    "upair(C,a,b,z) == a ∈ z ∧ b ∈ z ∧ (∀x[C]. x∈z ⟶ x = a ∨ x = b)"

definition
  upair_ax :: "(i⇒o) ⇒ o" **where**
    "upair_ax(C) == ∀x[C]. ∀y[C]. ∃z[C]. upair(C,x,y,z)"

We state the main result of this section in the context `M_extra_assms`.

lemma *pairing_axiom :*
  "one ∈ G ⟹ upair_ax(##M[G])"

Let x and y be elements in $M[G]$. By definition of the generic extension, there exist elements τ and ρ in M such that $x = val(G, \tau)$ and $y = val(G, \rho)$. We need to find an element in $M[G]$ that contains exactly these elements; for that we should construct a name $\sigma \in M$ such that $val(G, \sigma) = \{val(G, \tau), val(G, \rho)\}$.

The candidate, motivated by the definition of *check*, is $\sigma = \{\langle \tau, \mathrm{one}\rangle, \langle \rho, \mathrm{one}\rangle\}$. Our remaining tasks are to show

[8] The proof that $M[G]$ satisfies pairing only needs that M satisfies pairing.

(i) $\sigma \in M$, and

(ii) $val(G, \sigma) = \{val(G, \tau), val(G, \rho)\}$

By the implementation of pairs in *ZFC*, showing (i) involves using that the pairing axiom holds in M and the absoluteness of pairing thanks to M being transitive.

```
lemma pairs_in_M :
  " ⟦ a ∈ M ; b ∈ M ; c ∈ M ; d ∈ M ⟧ ⟹ {⟨a,c⟩,⟨b,d⟩} ∈ M"
```

Item (i) then follows because τ, ϱ and `one` belong to `M` (the last fact holds because `one`$\in$`P`, `P`$\in$`M` and `M` is transitive).

```
lemma sigma_in_M :
  " one ∈ G ⟹ τ ∈ M ⟹ ϱ ∈ M ⟹ {⟨τ,one⟩,⟨ϱ,one⟩} ∈ M"
by (rule pairs_in_M,simp_all add: upair_ax_def one_in_M)
```

Under the assumption that `one` belongs to the set `G`, (ii) follows from `def_val` almost automatically:

```
lemma valsigma :
  "one ∈ G ⟹ {⟨τ,one⟩,⟨ϱ,one⟩} ∈ M ⟹
   val(G,{⟨τ,one⟩,⟨ϱ,one⟩}) = {val(G,τ),val(G,ϱ)}"
```

6 Conclusions and future work

There are several technical milestones that have to be reached in the course of a formalization of the theory of forcing. The first one, and most obvious, is the bulk of set- and meta-theoretical concepts needed to work with. This pushed us, in a sense, into building on top of Isabelle/ZF, since we know of no other development in set theory of such depth (and breadth). In this paper we worked on setting the stage for the work with generic extensions; in particular, this involves some purely mathematical results, as the Rasiowa-Sikorski lemma.

Other milestones in this formalization project involve

(i) the definition of the forcing relation,

(ii) proving the Fundamental Theorem of forcing (that relates truth in M to that in $M[G]$), and

(iii) using it to show that $M[G] \models ZFC$.

The theory is very modular and this is witnessed by the fact that the last goal does not depend on the proof of the Fundamental Theorem nor on the definition of the forcing relation. Our next task will be to obtain the last goal in that enumeration.

To this end, we will develop an interface between Paulson's relativization results and countable models of *ZFC*. This will show that every ctm M is closed under well-founded recursion and, in particular, that contains names for each of its elements. Consequently, the proof of $M \subseteq M[G]$ will be complete. A landmark will be to prove the Axiom Scheme of Separation (the first that needs to use the machinery of forcing nontrivially). As a part of the new formalization, we will provide Isar versions of the longer applicative proofs presented in this work.

Acknowledgement

We'd like to thank the anonymous referees for reading the paper carefully and for their detailed and constructive criticism.

References

[1] Ballarin, C., *Tutorial to locales and locale interpretation*, in: *Contribuciones científicas en honor de Mirian Andrés Gómez*, Universidad de La Rioja, 2010, pp. 123–140.

[2] Cohen, P., *The independence of the continuum hypothesis*, Proc. Nat. Acad. Sci. U.S.A. **50** (1963), pp. 1143–1148.

[3] Gödel, K., "The Consistency of the Continuum Hypothesis," Annals of Mathematics Studies, no. 3, Princeton University Press, Princeton, N. J., 1940, 66 pp.

[4] Gonthier, G., *Formal proof—the four-color theorem*, Notices Amer. Math. Soc. **55** (2008), pp. 1382–1393.

[5] Hales, T., M. Adams, G. Bauer, T. D. Dang, J. Harrison, L. T. Hoang, C. Kaliszyk, V. Magron, S. Mclaughlin, T. T. Nguyen and et al., *A formal proof of the Kepler conjecture*, Forum Math. Pi **5** (2017), pp. e2, 29.
URL http://dx.doi.org/10.1017/fmp.2017.1

[6] Kunen, K., "Set Theory," Studies in Logic, College Publications, 2011, second edition, revised edition, 2013.

[7] Moschovakis, Y., "Notes on Set Theory," Springer Texts in Electrical Engineering, Springer-Verlag, 1994.

[8] Paulson, L. C., *Set Theory for Verification. II: Induction and Recursion*, Journal of Automated Reasoning **15** (1995), pp. 167–215.

[9] Paulson, L. C., *A fixedpoint approach to (co) inductive and (co) datatype definitions.*, in: *Proof, Language, and Interaction*, 2000, pp. 187–212.

[10] Paulson, L. C., *The relative consistency of the axiom of choice mechanized using Isabelle/ZF*, LMS Journal of Computation and Mathematics **6** (2003), pp. 198–248.

[11] Paulson, L. C., *ALEXANDRIA: Large-scale formal proof for the working mathematician*, Webpage (2017 — accessed September 2018), eC Project: `https://bit.ly/2Nb26ys`.

[12] Paulson, L. C., *Isabelle's logics: FOL and ZF*, Technical report, Computer Laboratory, University of Cambridge (2017).

[13] Paulson, L. C. and K. Grabczewski, *Mechanizing set theory*, J. Autom. Reasoning **17** (1996), pp. 291–323.
URL https://doi.org/10.1007/BF00283132

[14] Quirin, K., "Hypothèse du continu en théorie des types," Master's thesis, l'Université Nantes Angers Le Mans (2013).

[15] Rubin, H. and J. Rubin, "Equivalents of the Axiom of Choice, II," Studies in Logic and the Foundations of Mathematics, Elsevier Science, 1985.
URL https://www.elsevier.com/books/equivalents-of-the-axiom-of-choice-ii/rubin/978-0-444-87708-6

[16] Simpson, C., *Computer theorem proving in math*, Letters in Mathematical Physics **69** (2004), pp. 287–315.
URL http://dx.doi.org/10.1007/s11005-004-0607-9

[17] Univalent Foundations Program, T., "Homotopy Type Theory: Univalent Foundations of Mathematics," https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[18] Wenzel, M., *Isar - A generic interpretative approach to readable formal proof documents*, in: Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin-Mohring and L. Théry, editors, *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99, Nice, France, September, 1999, Proceedings*, Lecture Notes in Computer Science **1690** (1999), pp. 167–184.
URL https://doi.org/10.1007/3-540-48256-3_12

[19] Wenzel, M., L. C. Paulson and T. Nipkow, *The Isabelle framework*, in: *Theorem proving in higher order logics*, Lecture Notes in Comput. Sci. **5170**, Springer, Berlin, 2008 pp. 33–38.
URL http://dx.doi.org/10.1007/978-3-540-71067-7_7

[20] Zhan, B., *Formalization of the fundamental group in untyped set theory using auto2*, in: M. Ayala-Rincón and C. A. Muñoz, editors, *Interactive Theorem Proving* (2017), pp. 514–530.