CHAIN BOUNDING AND THE LEANEST PROOF OF ZORN'S LEMMA

GUILLERMO L. INCATASCIATO AND PEDRO SÁNCHEZ TERRAF

ABSTRACT. We present an exposition of the *Chain Bounding Lemma*, which is a common generalization of both Zorn's Lemma and the Bourbaki-Witt fixed point theorem. The proofs of these results through the use of Chain Bounding are amongst the simplest ones that we are aware of. As a by-product, we show that for every poset P and a function f from the powerset of P into P, there exists a maximal well-ordered chain whose family of initial segments is appropriately closed under f.

We also provide a "computer formalization" of our main results using the Lean proof assistant.

1. INTRODUCTION

This paper grew out of the search of an elementary proof of Zorn's Lemma. One such proof was obtained by the first author, which is similar to the one by Lewin [6].

After a careful examination, the authors realized that the method of proof actually yielded a pair of new, similar principles: *Chain Bounding* and the *Unbounded Chain Lemma*, which state the impossibility of finding strict upper bounds of linearly ordered subsets of posets. The first one is more fundamental, since it does not depend on the Axiom of Choice (AC); but coupled with the latter, it implies the second one and then Zorn's Lemma. Chain Bounding also implies the Bourbaki-Witt fixed point theorem; all these results are expounded in Section 3.

The original proof of Chain Bounding proceeded by contradiction, where a few relevant concepts were defined; this proof is essentially the one that appears in Appendix A, where it is used to show Zorn's Lemma in a self-contained manner. We realized that it was better instead to present those concepts ("good chains" and their comparability) independently to obtain positive results. These appear in Section 2, and Chain Bounding is now proved as a consequence of its main Theorem 4, the existence of a greatest good chain. Nevertheless, the main advantage of Chain Bounding in comparison to Theorem 4 is its straightforward statement and consequences, which make it more appealing as a "quotable principle".

We also include, in Section 4, a brief description of a "computer formalization" of our results in the Lean proof assistant.

Universidad Nacional de Córdoba. Facultad de Matemática, Astronomía, Física y Computación.
Centro de Investigación y Estudios de Matemática (CIEM-FaMAF), Conicet. Córdoba. Argentina.
Supported by Secyt-UNC project 33620230100751CB and Conicet PIP project 11220210100508CO.

2. The greatest good chain

We introduce some notation. Let P be a poset and $C \subseteq P$; we say that $s \in P$ is a *strict upper bound* of C if $\forall c \in C,\ c < s$. Furthermore, if $S \subseteq C$, we say that S is an *initial segment* of C ("$S \sqsubseteq C$") if for all $x \in C$, $x \leqslant y \in S$ implies $x \in S$. We will usually omit the word "initial" and simply say "S is a segment of C". The strict version of the segment relation is denoted by $S \sqsubset C$, that is, $S \sqsubseteq C$ and $S \neq C$. Finally, $\mathscr{P}(X)$ denotes the powerset of the set X.

Definition 1. Let $g : \mathscr{P}(P) \to \mathscr{P}(P)$ be given. We say that a chain $C \subseteq P$ is *good for g* if for all $S \sqsubset C$, $S \sqsubset g(S) \sqsubseteq C$.

When understood from the context, we omit "for g". We have the following key result.

Lemma 2 (Comparability)**.** *Let P be a poset and $g : \mathscr{P}(P) \to \mathscr{P}(P)$. If C_1 and C_2 are good chains, one is a segment of the other.*

Proof. Let $\mathscr{S}$ be the family of mutual segments of both C_1 and C_2. Hence $\bigcup\mathscr{S}$ is also a mutual segment. If $\bigcup\mathscr{S}$ is different from both C_1 and C_2, then $g(\bigcup\mathscr{S})$ should be a mutual segment since both are good; but this contradicts the fact that $g(\bigcup\mathscr{S}) \nsubseteq \bigcup\mathscr{S}$. □

Lemma 3. *The union of a family $\mathscr{F}$ of good chains is a good chain.*

Proof. The union $U := \bigcup\mathscr{F}$ is a chain by Comparability.

Note that every good chain D such that $D \subseteq U$ is a segment of U: Suppose that $c \in U$ and $c < d \in D$. Then $c \in C$ for some good $C \in \mathscr{F}$. If C is a segment of D, we have $c \in D$ and are done. Otherwise, the converse relation holds by Comparability and then we also have $c \in D$.

We will see that U is good. Let $S \subsetneq U$ be a proper segment of U. Then, there exists $d \in U$ such that $\forall c \in S,\ c < d$. Let D be a good chain such that $d \in D$. Since D is a segment of U, all those c belong to D. We conclude $S \subseteq D$, and since S is a segment of U, it is a segment of D and it is proper because $d \in D \setminus S$. Then $g(S)$ is segment of D, and hence $g(S)$ is a segment of U. □

By considering the union of *all* good chains, we readily obtain:

Theorem 4 (Greatest Good Chain)**.** *Let P be a poset and $g : \mathscr{P}(P) \to \mathscr{P}(P)$. The family of all good chains has a maximum under inclusion.* □

3. Chain Bounding and applications

We now present the main character of this work.

Lemma 5 (Chain Bounding)**.** *Let P be a poset. There is no assignment of a strict upper bound to each chain in P.*

Proof. Assume, by way of contradiction, that $f(C)$ is a strict upper bound of C for each chain $C \subseteq P$. Hence C is a proper segment of $g(C) := C \cup \{f(C)\}$; extend this g arbitrarily to the rest of the subsets of P. By Theorem 4, there exists a greatest good chain U for g. But this is a contradiction, since $g(U)$ is easily seen to be a good chain, but $g(U) \nsubseteq U$. □

The wording of the Chain Bounding Lemma is a bit awkward, since it is actually a negation. A more natural statement is the following:

Lemma 6 (Unbounded Chain)**.** *Assume AC. For every poset P there exists a chain $C \subseteq P$ with no strict upper bound.*

Proof. By way of contradiction, assume that for every chain $C \subseteq P$ there exists a strict upper bound. Using AC, let f assign to each C such a bound. But this contradicts Chain Bounding. □

Nevertheless, it should be noted that any proof of this last result requires an application of AC (Corollary 8), unlike Chain Bounding.

We now turn to applications. The first one is possibly the simplest proof of Zorn's Lemma (obviously taking into account the proof of the lemmas proved so far).

Corollary 7 (Zorn)**.** *If a poset P contains an upper bound for each chain, it has a maximal element.*

Proof. By the Unbounded Chain Lemma, take $C \subseteq P$ without strict upper bounds. Then any upper bound of C must be maximal in P. □

A direct, self-contained proof of Zorn's Lemma condensing all the ideas discussed up to this point appears in Appendix A below. This includes some simplifications that also apply to a direct proof of Chain Bounding; for instance, the definition of good chain is a bit shorter and one only needs to show that the union of all good chains is good.

Since Zorn's Lemma implies AC, we immediately have.

Corollary 8. *The Unbounded Chain Lemma is equivalent to AC over Zermelo-Fraenkel set theory.* □

Kunen points out in [4] that, for those not familiar with set theory, it may not be clear why Zorn's Lemma should be true, since the best-known proofs make use of ordinals and transfinite recursion. Our highest hope is that after seeing our proof, the old joke turns into *"AC is obviously true, the Well-Ordering Theorem is obviously false, and Zorn's Lemma... holds by Chain Bounding!"*.

Lewin, in [6], provides a very short proof without the need for ordinals or recursion, but making use of well-ordered chains. In [5], Lang presents a proof using the Bourbaki-Witt fixed point theorem of order theory, without even employing the concept of well-ordered set. Finally, Brown [1] gives a beautiful and simple proof inspired by Lang's but without the need for Bourbaki-Witt. This proof is slightly indirect since it actually proves the Hausdorff Maximal Principle and considers "closed" subsets in the poset of chains of the original poset ordered by inclusion.

The proof we presented here is not as short as Lewin's but it is more elementary since there is no use of (the basic theory of) well-orders in an explicit way. The main difference in method that allows to avoid them is to generalize his definition of "conforming chains" by considering general initial segments instead of *principal* ones (i.e., of the form $\{x \in P \mid x \leqslant p\}$ for some $p \in P$). This move allows to use the stronger expressiveness achieved by talking of general segments (indirectly referring to the powerset of P), but avoids talking of "second order" chains (i.e., chains in the poset of chains).

In spite of this simplification, it should be strongly emphasized the fundamental character of the concepts of well-order and well-foundedness in general. These are unavoidable in a sense; actually good chains for functions that add at most one element (such as the one in the proof of Chain Bounding) are well-ordered:

Proposition 9. *Let $(P,<)$ be poset, $f : \mathscr{P}(P) \to P$ and let $g(C) := C \cup \{f(C)\}$. Every good chain for g is well-ordered by $<$.*

Proof. We leave to the reader the verification of the fact, under the assumptions, that every segment D of a good chain is good.

Let C be a good chain and assume $X \subseteq C$ is nonempty. Let S be the set of strict *lower* bounds of X in C. Since $X \neq \varnothing$, S is a proper segment of C, and goodness ensures that $S \neq g(S) = S \cup \{f(S)\}$ is also a segment of C. Since $f(S) \notin S$, there is some $x \in X$ such that $x \leqslant f(S)$. We claim that any such x must be equal to $f(S)$ and hence it is the minimum element of X. For this, consider the good subchain $D := \{c \in C \mid c \leqslant x\}$ of C. Since S is likewise a proper segment of D, $f(S) \in g(S) \subseteq D$ and hence we obtain the claim. □

Moreover, it can be shown that a chain of a poset P is well-ordered if and only it is a good chain for some g as above.

Our second application is the aforementioned fixed point theorem.

Corollary 10 (Bourbaki-Witt)**.** *Let P be a non-empty poset such every chain $C \subseteq P$ has a least upper bound. If $h : P \to P$ satisfies $x \leqslant h(x)$ for all $x \in P$, then h has a fixed point, i.e., there is some $x \in P$ such that $x = h(x)$.*

Proof. Assume by way of contradiction that $x < h(x)$ for all $x \in P$. But then $f(C) := h(\sup C)$ immediately contradicts Chain Bounding. □

Note that the greatest good chain for $C \overset{g}{\longmapsto} C \cup \{h(\sup C)\}$ is the least complete subposet of P closed under h, and its ordinal length is (the successor of) the number of iterations of h needed to reach its least fixed point starting from the bottom element of P.

It is relevant here that Chain Bounding does not depend on AC, since Bourbaki-Witt can be proved without using it. This is another reason why we consider Chain Bounding the central item of this work.

4. A computer formalization in Lean

Although Lewin's proof is short, it could be debatable whether it is more straightforward for a general audience, as seen in the answers to [8] and [3].

A similar doubt could arise with our combo Chain-Bounding/Zorn: Some tricky details could still have been swept under the rug—or, even worse, some mistake.

The "Formal Methods" community in Computer Science developed modern programming technologies, *proof assistants* (PAs), that allow to write a mathematical proof in a formal language and the computer can then verify it for correctness. This is exactly the tool to assess the worries of the last paragraph.

The second author and collaborators discuss PAs in [2], where the reader will also find many pointers to other papers on the subject. Our formalization, which is available at

`https://github.com/sterraf/ChainBounding`,

is written using the PA and programming language *Lean* [7]. The website

`https://leanprover-community.github.io/learn.html`

has many indications on how to start using Lean, and the online forum

`https://leanprover.zulipchat.com/`

fosters the growing community of users and is very welcoming to newcomers.

The formalization of the main results of the paper takes up 420 lines of code. For the remainder of this section, we will focus on explaining only a fraction of the details involved (and in particular, some notations will not be dealt with); our main objective is to make a point that it is possible to translate mathematical reasoning into the computer, in a way that at least partially resembles the way it is done on paper. We hope that the reader's curiosity will be sufficiently motivated in order to visit the mentioned resources and to learn more about formalization and Lean.

We start our Lean file by *importing* basic results on chains, and the definition of complete partial orders (which appear in the Bourbaki-Witt Theorem).

```
import Mathlib.Order.Chain
import Mathlib.Order.CompletePartialOrder

variable {α : Type*}
```

The last line above indicates that we will be talking about a "type" α (which, in the type theory of Lean roughly corresponds to a set, or perhaps more appropriately, a set *underlying* some structure). Greek letters are commonly used for types, and here this α will replace our P from above.

We highlight some of the basic definitions. For instance, "S is a (proper) segment of C" is defined in the following way:

```
def IsSegment [LE α] (S C : Set α) : Prop := S ⊆ C ∧ ∀ c ∈ C, ∀ s ∈ S, c
    ⩽ s → c ∈ S

def IsPropSegment [LE α] (S C : Set α) : Prop := IsSegment S C ∧ S ≠ C
```

The arguments S and C appear declared as belonging to the powerset of α, which in Lean is written as `Set` α. The declaration in square brackets is an *implicit* argument stating that α belongs to the class of types having the $\leqslant$ notation defined (which is the bare minimum to be able to interpret the right hand side). After a few more lines, the declaration `variable [PartialOrder α]` states that we will be assuming a partial order structure on α.

We can actually set up infix notation for `IsSegment` and `IsPropSegment` in order to be able to write expressions as $S \sqsubseteq C$ and $S \sqsubset C$, as shown below.

After a new concept is introduced, a customary requisite is to write some extremely basic lemmas which allow one to work with it. These are referred to by the name *API*, an acronym for "application programming interface", a concept that comes from Computer Science. In our formalization, part of the API comprises all the possible transitivity lemmas involving $\sqsubseteq$, $\sqsubset$, or both.

We describe the formalization of the fact, used at the beginning of the proof of Lemma 2, that the union of a family of segments is a segment. The formalized statement is the following (where $\bigcup_0$ denotes the operator of union of a family), and the `by` keyword signals the start of the (tactic) proof:

```
lemma sUnion_of_IsSegment {F : Set (Set α)} (hF : ∀M ∈ F, M ⊑ C) : ⋃₀ F
    ⊑ C := by
```

Since $\bigcup_0 F \sqsubseteq C$ is defined by a conjunction, its justification is *constructed* by providing proofs for each conjunct. Each of those proofs appear indented and signaled by "·" below. We will analyze the first sub-proof line by line.

```
constructor
· intro s sInUnionF
  obtain ⟨M, MinF, sinM⟩ := sInUnionF
  exact (hF M MinF).1 sinM
· intro c cinC s sInUnionF cles
  obtain ⟨M, MinF, sinM⟩ := sInUnionF
  exact ⟨M, MinF, (hF M MinF).2 c cinC s sinM cles⟩
```

Right after writing `constructor` and the subsequent dot, the VS Code editor echoes:

```
α : Type u_1
inst † : PartialOrder α
C : Set α
F : Set (Set α)
hF : ∀ M ∈ F, M ⊑ C
⊢ ⋃₀ F ⊆ C
```

This "InfoView" lists all terms available to work (hypotheses are also included as "`Prop`ositional" terms), and the current *goal* (which, for this sub-proof, is the inclusion on the last line).

The natural way of producing a proof of that inclusion (defined by `∀ ⦃s⦄, s ∈ ⋃₀ F → s ∈ C`), is to *introduce* two new variables named `s` and `sInUnionF`,

```
· intro s sInUnionF
```

whose types ("α" and "`s ∈ ⋃₀ F`", respectively) are deduced from the previous goal. Now the InfoView turns into

```
α : Type u_1
inst † : PartialOrder α
C : Set α
F : Set (Set α)
hF : ∀ M ∈ F, M ⊑ C
s : α
sInUnionF : s ∈ ⋃₀ F
⊢ s ∈ C
```

From `sInUnionF`, which states by definition that `s` belongs to some element of `F`, we obtain such an element `M` and further terms/hypothesis that state the relations among them with the next line (where the "`:=`" can be read as "from"):

```
obtain ⟨M, MinF, sinM⟩ := sInUnionF
```

After this tactic, the propositional variables `MinF` and `sinM` state that `M ∈ F` and `s ∈ M`, respectively. Finally, we combine all the elements available by using some of the benefits of the type-theoretic framework:

- Logical constructs like implications and universal quantifiers behave as functions. For instance, the hypothesis `hF` (of type `∀ M, M ∈ F → M ⊑ C`) can be fed with the term `M` to obtain the implication `hf M` (having type `M ∈ F → M ⊑ C`) and the latter can be applied to `MinF` (a term for the antecedent) to obtain a term `hf M MinF` for the consequent `M ⊑ C`.
- The conjunction behaves as a Cartesian product, where components correspond to each conjunct. Hence the first component `(hf M MinF).1` is a term justifying `M ⊆ C = ∀ ⦃s⦄, s ∈ M → s ∈ C`.

By applying the last term obtained to `sinM`, we obtain `exact`ly what we were looking for, and the sub-proof ends.

```
    exact (hF M MinF).1 sinM
```

For the definition of goodness, we declare our own class consisting of types supporting a partial order, and we add an otherwise unspecified g. A special type of comment (a *docstring*) describes the concept introduced:

```
/--
A partial order with an *expander* function from subsets to subsets. In
    main applications, the
expander actually returns a bigger subset.
-/
class OrderExpander (α : Type*) [PartialOrder α] where
  g : Set α → Set α
```

Assuming the appropriate structures on α we are finally able to write down the definition.

```
variable [PartialOrder α] [OrderExpander α]

def Good (C : Set α) := IsChain ( · ≤ ·) C ∧ ∀ {S}, S ⊏ C → S ⊏ g S ∧ g
    S ⊑ C
```

A further class concerns partial orders with an "f",

```
class OrderSelector (α : Type*) [PartialOrder α] where
  f : Set α → α
```

and a statement that each type supporting it is an "instance" of `OrderExpander` in a canonical way. This is justified by presenting $C \overset{g}{\longmapsto} C \cup \{f(C)\}$ as the witness.

```
instance [PartialOrder α] [OrderSelector α] : OrderExpander α := ⟨fun C =>
    C ∪ {OrderSelector.f C}⟩
```

We skip directly to the statement of the Unbounded Chain Lemma,

```
lemma unbounded_chain [PartialOrder α] [Inhabited α] :
     ∃ C, IsChain ( · ≤ ·) C ∧ ¬ ∃ sb : α, ∀ a ∈ C, a < sb
```

which moreover assumes α to be nonempty (more precisely, that has a designated element) for simplicity, and the proof of Zorn's Lemma using it:

```
lemma zorn [PartialOrder α] [Inhabited α]
    (ind : ∀ (C : Set α), IsChain ( · ≤ ·) C → ∃ ub, ∀ a ∈ C, a ≤ ub) : ∃
    (x : α), IsMaximal x := by
  obtain ⟨C, chain, subd⟩ := unbounded_chain (α := α)
  push_neg at subd
  obtain ⟨ub, hub⟩ := ind C chain
  existsi ub
  intro z hz
  by_contra zneub
  obtain ⟨a, ainC, anltz⟩ := subd z
  exact anltz $ lt_of_le_of_lt (hub a ainC) $ lt_of_le_of_ne' hz zneub
```

We comment briefly on some of the tactics employed in this elementary proof. As before, `obtain` decomposes the statement of `unbounded_chain`, and in particular `subd` is a term asserting the truth of `¬ ∃ sb : α, ∀ a ∈ C, a < sb`. The tactic `push_neg` applies the De Morgan rules transforming it into `∀ (sb : α), ∃ a ∈ C, ¬a < sb`. The obtained upper bound `ub` for the (strictly) unbounded chain is presented as a witness to the existential quantifier of the conclusion by using the `existsi` tactic. After introducing variables `h` and `hz`, the `by_contra` tactic starts a proof by contradiction where the new hypothesis (the negation of the goal appearing immediately before) is stored in the variable `zneub`.

Appendix A. Zorn's on its own

For ease of reference, we streamline the arguments above to obtain a compact version of the proof.

Recall that for a poset P, $s \in P$ is a *strict upper bound* of $C \subseteq P$ if $\forall c \in C,\ c < s$; and that $S \subseteq C$ is a *segment* of C if for all $x \in C$, $x \leqslant y \in S$ implies $x \in S$.

Lemma (Zorn)**.** *If a poset P contains an upper bound for each chain, it has a maximal element.*

Proof. Assume by way of contradiction that $(P, \leqslant)$ does not have a maximal element. Hence, for every chain $C \subseteq P$ there exists a strict upper bound (otherwise, any upper bound of C would be maximal). Using the Axiom of Choice, let g assign $C \cup \{s\}$ to each chain $C \subseteq P$, where s is any such bound.

A chain $C \subseteq P$ is deemed to be *good* whenever

(*) If $S \neq C$ is a segment of C, then $g(S)$ also is.

We need the following property of good chains:

(*Comparability*) If C_1, C_2 are good, one is a segment of the other.

To prove it, let $\mathscr{S}$ be the family of mutual segments of both C_1 and C_2. Hence $\bigcup \mathscr{S}$ is also a mutual segment. If $\bigcup \mathscr{S}$ is different from both C_1 and C_2, then $g(\bigcup \mathscr{S})$ should be a mutual segment by (*); but this contradicts the fact that $g(\bigcup \mathscr{S}) \not\subseteq \bigcup \mathscr{S}$.

Let U be the union of all good chains, which is a chain by Comparability.

Note that every good chain D is a segment of U: Suppose that $c \in U$ and $c < d \in D$. Then $c \in C$ for some good C. If C were not a segment of D, then the converse relation would hold by Comparability and then we would also have $c \in D$.

We will see that U is good. Let $S \subsetneq U$ be a proper segment of U. Then, there exists $d \in U$ such that $\forall c \in S,\ c < d$. Let D be a good chain such that $d \in D$. Since D is a segment of U, all those c belong to D. We conclude $S \subseteq D$, and since S is a segment of U, it is a segment of D and it is proper because $d \in D \smallsetminus S$. Then $g(S)$ is segment of D, and hence $g(S)$ is a segment of U.

We reach a contradiction, since $g(U)$ is also a good chain, but $g(U) \not\subseteq U$. □

References

[1] K. Brown, Zorn's Lemma, Mathematics 6310, (2010). https://pi.math.cornell.edu/~kbrown/6310/zorn.pdf.

[2] E. Gunther, M. Pagano, P. Sánchez Terraf, M. Steinberg, The formal verification of the ctm approach to forcing, *Annals of Pure and Applied Logic* **175** (2024).

[3] C. Harumi, Proof of Zorn's lemma clarification, Mathematics Stack Exchange, (2019). https://math.stackexchange.com/q/3269781 (version: 2019-06-21).

[4] K. Kunen, "Set Theory", Studies in Logic, College Publications (2011), second edition. Revised edition, 2013.

[5] S. Lang, "Real and Functional Analysis", Graduate Texts in Mathematics, Springer New York, NY (2012).

[6] J. Lewin, A simple proof of Zorn's Lemma, *The American Mathematical Monthly* **98**: 353–354 (1991).

[7] L. de Moura, S. Ullrich, The Lean 4 theorem prover and programming language, in: A. Platzer, G. Sutcliffe (Eds.), Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings, Lecture Notes in Computer Science **12699**, Springer: 625–635 (2021).

[8] WillG, Question on a step of a simple proof of Zorn's Lemma by Lewin, Mathematics Stack Exchange, (2020). https://math.stackexchange.com/q/3616209(version: 2020-04-28).

Email address: guillermo.incatasciato@mi.unc.edu.ar

Email address: psterraf@unc.edu.ar